AD-A148 751

NAVAL OCEAN SYSTEMS CENTER, SAN DIEGO, CA
APSE INTERACTIVE MONITOR – USER'S MANUAL-ADE™
VERSION BY: TEXAS INSTRUMENTS INC.

1 OF 4
NOSC TD 1218
UNCLASSIFIED
JAN 1985

UNCLASSIFIED

NOSC

NAVAL OCEAN SYSTEMS CENTER San Diego, California 92152-5000

NOSC TD 1218

Technical Document 1218
January 1988

APSE Interactive Monitor

User's Manual-ADE™ Version

Texas Instruments Incorporated

UNITED STATES NAVY
NAVAL OCEAN SYSTEMS CENTER

ADMINISTRATIVE INFORMATION

This task was performed for the Ada Joint Program Office, Arlington, VA 22209. Work was done under contract N66001-82-C-0440 by Texas Instruments Incorporated, McKinney, TX 75069. The Contracting Officer's Technical Representative was P.A. Oberndorf, Software Engineering Technology Branch, Naval Ocean Systems Center, San Diego, CA 92152-5000.

Released by
R.A. Wasilausky, Head
Software Engineering
Technology Branch

Under authority of
J.A. Salzmann, Head
Information Systems
Division

LH

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited.
2b DECLASSIFICATION DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S): NOSC TD 1218
6a NAME OF PERFORMING ORGANIZATION: Texas Instruments Incorporated Equipment Group -- ACSL; 6b OFFICE SYMBOL (if applicable)	7a NAME OF MONITORING ORGANIZATION: Naval Ocean Systems Center
6c ADDRESS (City, State and ZIP Code): P.O. Box 801, M.S. 8007 McKinney, TX 75069	7b ADDRESS (City, State and ZIP Code): San Diego, CA 92152-5000
8a NAME OF FUNDING SPONSORING ORGANIZATION: Ada Joint Program Office; 8b OFFICE SYMBOL (if applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: N66001-82-C-0440
8c ADDRESS (City, State and ZIP Code): 1211 S. Fern Arlington, VA 22209	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	AGENCY ACCESSION NO
63226F	RDAF	RDAF	DN288 534

11 TITLE (Include Security Classification)
APSE INTERACTIVE MONITOR
User's Manual-ADE ™ Version

12 PERSONAL AUTHOR(S)
Texas Instruments Incorporated

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
	FROM ______ TO ______	January 1988	335

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB GROUP	Ada
			Ada Programming Support Environment (APSE)
			APSE Interactive Monitor (AIM)

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The APSE Interactive Monitor User's Manual provides a useful reference for the AIM regardless of the APSE in which it is to be used. The manual contains a glossary, a description of AIM, command reference information, AIM operation instructions, and a tutorial.

20 DISTRIBUTION AVAILABILITY OF ABSTRACT: ☐ UNCLASSIFIED UNLIMITED [✓ SAME AS RPT] ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL: P.A. Oberndorf	22b TELEPHONE (Include Area Code): (619) 553-4080; 22c OFFICE SYMBOL: Code 423

DD FORM 1473, 84 JAN — 83 APR EDITION MAY BE USED UNTIL EXHAUSTED. ALL OTHER EDITIONS ARE OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DD FORM 1473, 84 JAN

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PURPOSE

The Ada [Ada is a registered trademark of the U.S. Government, Ada Joint Program Office] Programming Support Environment (APSE) Interactive Monitor (AIM) User's Manual was written to provide a useful reference manual for the AIM regardless of the APSE in which it is to be used. The AIM is under development for the Naval Ocean Systems Center (NOSC), Contract N66001-82-C-0440.

1.2 SCOPE

The AIM User's Manual contains a glossary (Chapter 2), a description of the AIM (Chapter 3), command reference information (Chapter 4), AIM operating instructions, (Chapter 5), and a tutorial (Appendix A).

1.3 ORGANIZATION

The AIM User's Manual was written to accommodate any APSE in which the AIM is to be used. Only Chapter 5 and the appendix containing the tutorial need be changed for any APSE. Chapter 5 contains system specific operating instructions and the tutorial contain system specific examples which may require changes for each APSE. Chapter 4 of the manual is an AIM commands reference guide. Chapter 4, in combination with the appendix, covers most aspects of the AIM commands.

CHAPTER 2

GLOSSARY

2.1 ACRONYMS

BNF
    Backus-Naur Form

ADE
    Data Gene: Corporation's Ada Development Environment

AIE
    Ada Integrated Environment

AIM
    APSE Interactive Monitor

AIM CLI
    AIM Command Language Interpreter

ALS
    Ada Language System

APSE
    Ada Programming Support Environment

CPCI
    Computer Program Configuration Item.

KAPSE
    Kernel Ada Programing Support Environment

MAPSE
    Minimum Ada Programing Support Environment

NOSC	Naval Ocean Systems Center

2.2 DEFINITIONS

Ada Programming Support Environment (APSE)
The purpose of an APSE is to support the development and maintenance of Ada applications software throughout its life cycle, with particular emphasis on software for embedded computer applications. The principal features are the database, the interface and the toolset. It is structrued in levels:
level 0: hardware and host software as appropriate
level 1: KAPSE
level 2: MAPSE
level 3: APSE's which are constructed by extensions of the MAPSE to provide fuller support for particular applications or methodologies.

AIM Command Language Interpreter (AIM CLI)
The AIM subsystem concerned with the interpretation of user commands. The user commands include those specified directly to the AIM CLI by the user as well as special key sequences (such as the SWITCH_TO_AIM key).

AIM virtual terminal
The AIM subsystem concerned with the manipulation and interrogation of display terminals.

APSE Command Language Processor (CLP)
The APSE Command Language Processor is the program within the host APSE concerned with the interpretation of user commands at the APSE level.

APSE program
A program that can be executed in the hosting APSE and uses only KAPSE supplied services to perform its function.

area
A set of adjacent character positions in a visual display that are not necessarily on the same line.

Backus-Naur Form
A context-free grammer specification for the syntax of programming language.

character
A member of a set of elements that is used for the organization, control, or representation of data.

character echo
The act of re-transmitting a character upon receipt of it back to the entity that originally transmitted it.

character imaging device
A device that gives a visual representation of data in the form of graphic symbols using any technology, such as cathode ray tube or printer.

character stream
An unbounded sequence of ASCII characters.

character string
A bounded sequence of ASCII characters.

database file
A standard file in the APSE database.

display
The area for visual presentation of data on a character imaging device.

display terminal
A data communications device composed of a keyboard and a display screen (usually a cathode ray tube).

editing extent
The physical extent on the display screen where user editing of character data is to be allowed. Valid editing extent are the display screen, a line on the display, and a field within a line.

hardcopy terminal
A data communications device composed of a keyboard and a printer.

image
An image is an analog of the physical display device. The image is the entity that is mapped onto the display. Given a number of user defined images, only one at a time can be mapped onto the display. The other images exist and are updated asynchronously but are not mapped onto the display until the user requests it.

input pad
An APSE database file used to log a particular window's input. When a window's input pad is active, all input destined for the window is written to this file; the logging begins when the user activates the pad and continues until the pad is deactivated. The user can directly deactivate a window's input pad via a textual command, or may indirectly deactivate it by deleting the window or exiting the AIM.

interface
(1) A shared boundary. (2) the set of data passed between two or more programs or segments of programs, and the assumptions made by each program about how the other(s) operate. (3) The common boundary between software modules, between hardware devices, or between hardware and software modules. (4) When applied to a module, that set of assumptions made concerning the module by the remaining program or system in which it appears. Modules have control, data, and services interfaces. An interface can be represented by means of Ada package syntax and semantics.

Kernel Ada Programming Support Environment (KAPSE)
That level of an APSE which provides database communication and runtime support functions to enable the execution of an Ada program (including a MAPSE tool) and which presents a machine-independent portability interface.

keyboard
The keyboard is the physical input device.

line
A line is a set of adjacent character positions in a visual display that have the same vertical position.

Minimum Ada Programming Support Environment (MAPSE)
That level of an APSE which provides a minimal set of tools, written in Ada and supported by the KAPSE, which are both necessary and sufficient for the development and continuing support of Ada programs.

output pad
An APSE database file used to log a particular window's output. When a window's output pad is active, all window output destined for the terminal is written to this file; the logging begins when the user activates the pad and continues until the pad is deactivated. The user can directly deactivate a window's output pad via a textual command, or may indirectly deactivate it by deleting the window or exiting the AIM.

pipe
A logical connection between an output file of one program and an input file of another program.

qualified area
An area on the display screen that may have restrictions on the type of data that may be entered and whose boundaries are specified special qualifiers. See "qualifiers".

qualifiers
: Controls that define what is valid data that can be entered in specific areas on the display screen. Valid qualifiers include: graphics, numerics, all input, no input (protected), etc.

screen
: The area for visual presentation of data on any type of character imaging device, including printer and cathode ray tube device.

scroll mode terminal
: A display terminal that presents data by moving all the graphic symbols of the screen in one direction to make room for new data.

Selector
: An operation on an object that returns a value or attribute and makes no alteration of the state, structure, or value of the object.

SWITCH_TO_AIM key
: The keyboard sequence that causes the AIM to perform the SWITCH_TO_AIM function.

task
: An Ada program unit that operates in parallel with other program units.

terminal
: A data communications device consisting of a keyboard and a character imaging device.

Terminal Capabilities File
: The Terminal Capabilities File is an APSE database file which contains mappings from logical computer terminal characteristics to specific computer terminal characteristics. The Terminal Capabilities File can contain information that supports many different terminal types and makes.

transmit
: To send data as a data stream for purposes of information interchange.

transmitted data stream
: The stream of bit combinations sent by a character imaging device when it is induced to transmit for purposes of information interchange.

user terminal
: The terminal with which a user interacts in order to communicate with an APSE program.

viewport
: A viewport is the portion of the window displayed in the image.

viewport header
: A viewport header is a single highlighted line located at the top of a viewport.

window
: A window is an analog of the APSE program's view of the terminal.

CHAPTER 3

AIM SYSTEM CAPABILITIES

3.1 PURPOSE

The APSE Interactive Monitor (AIM) is a computer program that acts as an interface between a user of the APSE and the programs the user executes in the APSE.

3.2 GENERAL DESCRIPTIONS

The AIM allows a user to have multiple APSE programs executing while keeping their interactive inputs and outputs separate both logically and physically. Facilities are provided by a simple command language to supplement or replace the standard functions available through the APSE user interface in the area of terminal and program control.

3.3 FUNCTION PERFORMED

The AIM will interface with page mode computer terminals.

Page mode terminals transmit and receive characters one at a time. When a key is pressed on the keyboard, the character corresponding to that key is transmitted. When a character is received by the terminal, the character is displayed or performs a simple function such as carriage return or line feed. Additionally, cursor movement and screen editing capabilities such as cursor positioning and character and/or line insertion/deletion are provided.

3.3.1 Images

An AIM user defines structures called images, each of which is an analog of the user's display. As such, the length (number of lines) and width (number of character positions) attributes of an image are identical to that of the display. Note that only characters will be supported; no graphic support is provided. Any number of images may coexist at one time, and the user selects which image is "mapped onto" the display. Being "mapped onto" means that any changes to the information in an image are immediately reflected on the display. Only one image may be mapped onto the display at any given time.

3.3.2 Windows

An AIM user also defines structures called windows, a window being the analog of the APSE program's view of the terminal. The terminal output of the APSE program is intercepted by the AIM and directed to a structure called a window. There is exactly one window associated with each APSE program executing directly under the AIM.

3.3.3 Viewports

A window is mapped onto an image through a structure called a viewport. A viewport is a rectangular area within an image. The width of a viewport is equal to the width of the image. The length is user determined but can be no larger than the image length and no smaller than two lines (This includes a required line for the viewport header). As many viewports can be defined as will fit on any given image. The user defines the viewports by creating associations between images and windows and defining the relationships between them (position and length). Viewports on the same image are non-intersecting. Horizontal partitioning is supported; vertical partitioning is not. The space on an image that does not contain a viewport is considered dead space and no information is mapped onto it. A window may be associated with more than one viewport at the same time, but a specific window may only be associated with a specific image once.

3.3.3.1 Viewport Header -

Each viewport will have an identification line associated with it called a viewport header. The viewport header occupies one line at the top of each viewport (included as part of the viewport length).

The AIM provides viewport information in the form of a header line. The viewport header fields follow:

Description	Position	Length	What to expect
	1	1	blank
Image Name...............	2--15	14	Ada identifier
	16	1	blank
Window Name.......... ..	17--30	14	Ada identifier
	31	1	blank
Program Status..........	32--55	24	string
	56	1	blank
More Window Output......	57--70	14	string
	71	1	blank
Mode Flags.............	72--80	9	described below

The Mode Flags are as follows:

a. Suspended program output (S)

b. Pause on full window (F)

c. Input Pad (I)

d. Output pad (O)

e. Suspend output on SWITCH_TO_AIM command (A)

Note:

a. Text too large for its appropriate field will be truncated on the right in order to fit within the field boundaries.

b. The viewport headers will be emphasized in the terminal display (whether that be highlighted, inverse video, underlined, etc.) if the terminal can support such emphasis.

3.3.4 Pre-Defined Images

Two predefined images (named AIM and MAIN) are created when the AIM program is invoked.

3.3.4.1 AIM -

The AIM image is a predefined image associated (through a viewport) with a predefined window also called AIM. The AIM image and the AIM window cannot be deleted, nor can they be disassociated since the AIM image communicates directly with the AIM command interpreter to perform the functions associated with the AIM processing. A special key sequence is provided to switch a user directly to this image.

3.3.4.2 MAIN -

The MAIN image is a predefined image associated (through a viewport) with a single predefined window also called MAIN. An APSE program is started and associated with this window upon invoking the AIM. This APSE program is normally the APSE command language processor. The MAIN image is the one initially connected to the user's terminal. The MAIN image and window are exactly like all others from this point on. They can be deleted or reassociated at any time.

3.3.5 Pads

An AIM user can at any time request the activation of a window's input pad, output pad, or both; each activation of a pad creates a unique database file. A pad may be activated or deactivated for each window separately. A summary of pad functionality follows:

a. From activation until the window is deleted or the user deactivates the pad(s) associated with the window, a pad(s) log all session input and/or output activity for a particular window.

b. Deleting a window which has a(n) active pad(s) causes the associated database file(s) to be closed.

c. Deactivating a pad(s) associated with a window results in the associated database file(s) being closed.

d. Once a pad has been closed, activating the pad again for the same window causes the creation of a new file.

e. When the AIM EXIT or ABORT command is used, all active pad(s) will be closed.

f. Pad file(s) which have been closed are accessible (environmentally dependent) by other APSE tools at a later date.

3.3.6 Terminal Capabilities File

The Terminal Capabilities file (TCF) is an APSE database file containing mappings from logical computer terminal characteristics to specific computer terminal characteristics. This file contains information that describes the common terminal functions in terms of device-specific character sequences. An initial set of terminal capabilities is provided with the AIM defining the capabilities of terminals produced by some major manufacturers. Capabilities of terminals not defined in the terminal capabilities file may be added by a system manager.

The terminal capabilities file contains one entry for each different terminal type. Each entry contains the terminal specific character codes and/or sequences needed to support all the required capabilities of the AIM virtual terminal.

CHAPTER 4

AIM COMMANDS

The first section of this chapter covers command variations. Within Command Variation, the areas of command recognition, command completion, parameter prompting, and implicit help are discussed. Command Variation is followed by sections describing the text commands and keystroke commands respectively. Each option of a Textual Command is explained separately. For example, the CREATE command has two different options (WINDOW or IMAGE) and each of these is explained in its own section.

4.1 COMMAND VARIATION

The AIM command language was designed to be very flexible. Since the language will be used both interactively (input from the terminal) and in "batch" mode (input from a command script), a language format that was easy to type and yet easy to read and understand was desired. Therefore, AIM commands may have two syntax forms: an Ada-like form or short form. Either form of the command may be used interchangeably during an AIM session. The AIM command interpreter recognizes minimal strings of an AIM command; for example, instead of typing "CREATE", the user may simply type "CR" which uniquely identifies the CREATE command.

The following sessions demonstrate the various possibilities for command input including command completion, command recognition, parameter prompting, and implicit help. These variations are NOT shown for every command but they do apply to all commands. The variations shown can be used at anytime when entering AIM commands.

Session Assumptions:

1. Data General Corporation's Ada Development Environment

2. a Page terminal

3. terminal display with 24 lines and 80 character columns

For the reader's convenience, the following conventions will be used in the presentation of this session:

1. ALL user responses will appear on the last nonblank line of the screen

2. the affect of executing a user's command will be presented in the next screen relative to when the command was issued

3. the comments below each screen will explain:

   a. the affect of executing the previous AIM command

   b. the contents of the current screen

   c. the new AIM command issued at the bottom of the present screen

The default key sequences that exist when the AIM is invoked from a ASCII Page Mode Terminal are as follows:

DEFAULT KEYSTROKES

ABORT_SCRIPT	F1
CLEAR_WINDOW	F2
NEXT_IMAGE	F3
NEXT_PAGE	F4
NEXT_VIEWPORT	F5
PREVIOUS_IMAGE	F6
PREVIOUS_VIEWPORT	F7
REDISPLAY_SCREEN	F8
RETURN_TO_PREVIOUS_IMAGE	F9
SWITCH_TO_AIM	F10

where "F#" represents a function key.

4.1.1 COMMAND RECOGNITION

The AIM provides four forms for commands:

a. Long Form (named parameter association)

b. Abbreviated Long form (mixed)

c. Short Form (positional)

d. Abbreviated Short form

The following session demonstrates each of these forms. The first screen shows a sample. It is followed by an actual AIM session.

SAMPLE

```
 AIM                AIM      AIM CLI       Running                      AF

AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WIN1 -- Long Form

              -- is identical to

AIM> CR OBJECT_TYPE => WINDOW WINDOW_NAME => WIN1    -- Abbreviated Long For

              -- is identical to

AIM> CREATE WINDOW WIN1                             -- Short Form

              -- is identical to

AIM> CR W WIN1                                      -- Abbreviated Short For



______________________________BOTTOM-OF-SCREEN_______________________________
```

The AIM session follows.

INVOKE AIM FROM APSE

```
-) AIM
```

```
________________________BOTTOM-OF-SCREEN________________________
```

Comments:

Command: AIM - invoke the AIM from the APSE.

SWITCH TO AIM IMAGE

```
__________________________________________________________________________
 MAIN              MAIN          FRCH:016          Running              AF

AOS/VS CLI REV 05.01.00.00          9-APR-85      13:26:43 {F10}




________________________________BOTTOM-OF-SCREEN__________________________
```

Comments: The AIM initializes the current image to be MAIN. To enter AIM textual commands the user must communicate through the AIM image to the AIM command interpreter.

Command: {F10} - Set the current image to be the AIM image by using the SWITCH_TO_AIM keystroke.

LONG FORM

```
 AIM              AIM       AIM CLI       Running                         AF
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WIN1
```

```
________________________________BOTTOM-OF-SCREEN________________________________
```

Comments: The cursor is now at the top of the AIM image. AIM commands can only be entered from the AIM image. Control has been passed to the AIM CLI executing in the AIM window.

Command: CREATE OBJECT TYPE => WINDOW WINDOW NAME => WIN1 - Create a window in the AIM environment named WIN1.

ABBREVIATED LONG FORM

```
 AIM              AIM              Running                                   AF
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WIN1
AIM> CR OBJECT_TYPE => WINDOW WINDOW_NAME => WIN2










______________________________BOTTOM-OF-SCREEN________________________
```

Comments: Show long (named) form of create command.

Command: CR OBJECT TYPE => WINDOW WINDOW NAME => WIN2 - Create a window in the AIM environment named WIN2.

SHORT FORM

```
 AIM               AIM       AIM CLI       Running                            AF
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WIN1
AIM> CR OBJECT_TYPE => WINDOW WINDOW_NAME => WIN2
AIM> CREATE WINDOW WIN3









________________________________BOTTOM-OF-SCREEN________________________________
```

Comments: The "CR" followed by named parameters is the abbreviated long form of create command.

Command: CREATE WINDOW WIN3 - Create the window WIN3.

ABBREVIATED FORM

```
 AIM                 AIM       AIM CLI       Running                         AF
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WIN1
AIM> CR OBJECT_TYPE => WINDOW WINDOW_NAME => WIN2
AIM> CREATE WINDOW WIN3
AIM> CR W WIN4
```

BOTTOM-OF-SCREEN

Comments: Show abbreviated short (positional) form of create command.

Command: CR W WIN4 - Create the window WIN4.

ENTER EXIT COMMAND

```
 AIM              AIM       AIM CLI       Running                         AF
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WIN1
AIM> CR OBJECT_TYPE => WINDOW WINDOW_NAME => WIN2
AIM> CREATE WINDOW WIN3
AIM> CR W WIN4
AIM> EXIT
```

```
______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: Show the minimal form of create command. All commands can be abbreviated.

Command: EXIT - Enter the exit command.

EXIT

```
 AIM                 AIM       AIM CLI       Running                         AF
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WIN1
AIM> CR OBJECT_TYPE => WINDOW WINDOW_NAME => WIN2
AIM> CREATE WINDOW WIN3
AIM> CR W WIN4
AIM> EXIT
-)







                                BOTTOM-OF-SCREEN
```

Comments: Exit the AIM environment and return to the ADE.

Command: None.

4.1.2 COMMAND COMPLETION

Command Completion is provided by the AIM. The command interpreter will complete any recognized keyword. This session demonstrates the command completion facility. The first screen is a sample. It is followed by an actual AIM session.

SAMPLE

```
 AIM              AIM        AIM CLI        Running                        AF

AIM> CR                                      -- User enters abbreviated command
                                                followed by a carriage return.
AIM> CREATE                                  -- AIM completes the command


and


AIM> CREATE I                                -- User enters abbreviated object
                                                followed by a carriage return.
AIM> CREATE IMAGE                            -- AIM completes the object




______________________________BOTTOM-OF-SCREEN______________________________
```

The AIM session follows.

INVOKE AIM FROM APSE

```
-) AIM
```

BOTTOM-OF-SCREEN

Comments:

Command: AIM - invoke the AIM from the APSE.

SWITCH TO AIM IMAGE

```
_________________________________________________________________________
 MAIN            MAIN           FRCH:016        Running                AF

AOS/VS CLI REV 05.01.00.00        9-APR-85     13:26:43 {F10}











__________________________BOTTOM-OF-SCREEN_______________________________
```

Comments: The AIM initializes the current image to be MAIN. To enter AIM textual commands the user must communicate through the AIM image to the AIM command interpreter.

Command: {F10} - Set the current image to be the AIM image by using the SWITCH_TO_AIM keystroke.

COMMAND COMPLETION

```
 AIM                AIM       AIM CLI       Running                          AF
AIM> CR





                              BOTTOM-OF-SCREEN
```

Comments: The cursor is now at the top of the AIM image. AIM commands can only be entered from the AIM image. Control has been passed to the AIM CLI executing in the AIM window.

Command: CR - enter enough letters to uniquely identify the CREATE command.

COMMAND COMPLETION

```
 AIM               AIM         AIM CLI        Running                        AF
AIM> CREATE I
















______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The AIM CLI completes the command when a carriage return is entered; "CR" is now "CREATE".

Command: CREATE I - enter enough characters to uniquely identify image.

COMMAND COMPLETION

```
 AIM               AIM     AIM CLI        Running                             AF
AIM> CREATE IMAGE IMAGE_1









_______________________________BOTTOM-OF-SCREEN_______________________________
```

Comments: The AIM CLI completes the object when a carriage return is entered; "I" is now "IMAGE".

Command: CREATE IMAGE IMAGE_1 - Enter the image name.

ENTER EXIT COMMAND

```
 AIM            AIM      AIM CLI       Running                     AF
AIM> CREATE IMAGE  IMAGE_1
AIM> EXIT





                         BOTTOM-OF-SCREEN
```

Comments: The image name has been entered.

Command: EXIT - Exit the AIM.

EXIT

```
 AIM              AIM        AIM CLI        Running                        AF
AIM> CREATE IMAGE  IMAGE_1
AIM> EXIT
-)









______________________________BOTTOM-OF-SCREEN____________________________
```

Comments: Exit the AIM.

Command: None.

4.1.3 PARAMETER PROMPTING

The AIM will prompt for parameters missing from a recognized command.

SAMPLE

```
 AIM               AIM       AIM CLI       Running                        AF
AIM> CREATE                                     -- User Input
_OBJECT_TYPE =>                                 -- System Response is to
One of the following tokens expected:              prompt for object type.
IMAGE             WINDOW




_OBJECT_TYPEAIM> IMAGE                         -- User Inputs object type
_IMAGE_NAME => IM1                             -- System Response is to
                                                  prompt for the name.









______________________________BOTTOM-OF-SCREEN______________________________
```

The AIM session follows:

INVOKE AIM FROM APSE

```
) AIM
```

BOTTOM-OF-SCREEN

Comments:

Command: AIM - invoke the AIM from the APSE.

SWITCH TO AIM IMAGE

```
________________________________________________________________________
 MAIN                MAIN          FRCH:016        Running             AF

AOS/VS CLI REV 05.01.00.00         9-APR-85      13:26:43 {F10}










__________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The AIM initializes the current image to be MAIN. To enter AIM textual commands the user must communicate through the AIM image to the AIM command interpreter.

Command: {F10} - Set the current image to be the AIM image by using the SWITCH_TO_AIM keystroke.

PARAMETER PROMPTING

```
 AIM               AIM        AIM CLI        Running                              AF
AIM> CREATE















                              BOTTOM-OF-SCREEN
```

Comments: The cursor is now at the top of the AIM image. AIM commands can only be entered from the AIM image. Control has been passed to the AIM CLI executing in the AIM window.

Command: CREATE - Enter the create command.

PARAMETER PROMPTING

```
 AIM            AIM        AIM CLI        Running                        AF
AIM> CREATE
_OBJECT_TYPE => IMAGE




_________________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The AIM prompt appears and the AIM header is at the top of the screen. Enter the create command and hit return to be prompted.

Command: IMAGE - Create an image.

PARAMETER PROMPTING

```
 AIM            AIM       AIM CLI        Running                          AF
AIM> CREATE
_OBJECT_TYPE => IMAGE
_IMAGE_NAME => IM1









BOTTOM-OF-SCREEN
```

Comments: The user is now prompted to enter an object type. Enter either window or image.

Command: `_IMAGE_NAME => IM1`

ENTER EXIT COMMAND

```
 AIM              AIM         AIM CLI        Running                             AF
AIM> CREATE
_OBJECT_TYPE => IMAGE
_IMAGE_NAME => IM1
AIM> EXIT
```

BOTTOM-OF-SCREEN

Comments: The user is now prompted for the image name. The image IM1 is created.

Command: EXIT

EXIT

```
 AIM                AIM      AIM CLI       Running                      AF
AIM> CREATE
_OBJECT_TYPE => IMAGE
_IMAGE_NAME => IM1
AIM> EXIT
-)
```

BOTTOM-OF-SCREEN

Comments: Exit the AIM.

Command: None.

4.1.4 IMPLICIT HELP

The AIM provides an implicit help function which is requested by "?". The "?" may appear anywhere within an AIM command. The following is a demonstration of the Implicit Help function. The first screen is a sample. It is followed by an actual AIM session.

SAMPLE

```
 AIM              AIM       AIM CLI        Running                           AF
AIM> D?
One of the following tokens expected:
DEFINE                  DELETE
DISASSOCIATE
AIM> D                                              -- AIM Prompts



AIM> DEL ?                                          -- User enters "EL" after the
One of the following tokens expected:                  "D"
IMAGE                   WINDOW
OBJECT_TYPE
AIM> DEL                                            -- Delete Image or Window



AIM> DEL I ?
IMAGE_NAME
AIM> DEL I                                          -- Image name requested



_______________________________BOTTOM-OF-SCREEN________________________________
```

The AIM session follows.

INVOKE AIM FROM APSE

```
) AIM
```

```
______________________BOTTOM-OF-SCREEN______________________
```

Comments:

Command: AIM - invoke the AIM from the APSE.

SWITCH TO AIM IMAGE

```
MAIN          MAIN        FRCH:016       Running.                      AF

AOS/VS CLI REV 05.01.00.00        9-APR-85      13:26:43 {F10}











___________________________BOTTOM-OF-SCREEN________________________________
```

Comments: The AIM initializes the current image to be MAIN. To enter AIM textual commands the user must communicate through the AIM image to the AIM command interpreter.

Command: {F10} - Set the current image to be the AIM image by using the SWITCH_TO_AIM keystroke.

IMPLICIT HELP

```
 AIM              AIM        AIM CLI        Running                            AF
AIM> CREATE IMAGE IM1




                                  BOTTOM-OF-SCREEN
```

Comments: The cursor is now at the top of the AIM image. AIM commands ca only be entered from the AIM image. Control has been passed to the AIM CL executing in the AIM window.

Command: CREATE IMAGE IM1 - Create the image, IM_1

IMPLICIT HELP

```
 AIM              AIM     .  AIM CLI         Running                          AF
AIM> CREATE IMAGE IM1
AIM> D?









BOTTOM-OF-SCREEN
```

Comments: Create an image to delete later in the session.

Command: D? - see what commands start with a "D"

IMPLICIT HELP

```
 AIM              AIM       AIM CLI        Running                          AF
AIM> CREATE IMAGE IM1
AIM> D?
One of the following tokens expected:
DELETE               DEFINE
DISASSOCIATE
AIM> DEL ?
```

```
______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The user enters a single valid character and requests help.

Command: EL ?

IMPLICIT HELP

```
 AIM            AIM       AIM CLI        Running                          AF
AIM> CREATE IMAGE IM1
AIM> D?
One of the following tokens expected:
DEFINE              DELETE
DISASSOCIATE
AIM> DEL ?
One of the following tokens expected:  One of the following:
IMAGE               WINDOW
OBJECT_TYPE
AIM> DEL I ?
```

```
________________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The commands that start with a "D" are listed. Enter enough characters to uniquely identify the DELETE command and then request help on it.

Command: EL I ? Delete an image

IMPLICIT HELP

```
 AIM                AIM       AIM CLI        Running                      AF
AIM> CREATE IMAGE IM1
AIM> D?
One of the following tokens expected:
DEFINE              DELETE
DISASSOCIATE
AIM> DEL ?
One of the following tokens expected:
IMAGE               WINDOW
OBJECT_TYPE
AIM> DEL I ?
One of the following tokens expected:
IMAGE_NAME          IDENTIFIER
AIM> DEL I IM1
```

________________________BOTTOM-OF-SCREEN________________________

Comments: The DELETE command was recognized. The user can delete an image or a window. The user responds by indicating the minimum number of characters for an image.

Command: IM1 - Delete the image created earlier.

ENTER EXIT COMMAND

```
 AIM             AIM      AIM CLI       Running                     AF
AIM> CREATE IMAGE IM1
AIM> D?
One of the following tokens expected:
DEFINE               DELETE
DISASSOCIATE
AIM> DEL ?
One of the following tokens expected:
IMAGE                WINDOW
OBJECT_TYPE
AIM> DEL I ?
One of the following tokens expected:
IMAGE_NAME           IDENTIFIER
AIM> DEL I IM1
AIM> EXIT








________________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The AIM CLI responds with the images that can be deleted. The user deletes the image created earlier.

Command: EXIT - Exit the AIM.

EXIT

```
 AIM                AIM                Running                                       AF
AIM> CREATE IMAGE IM1
AIM> D?
One of the following tokens expected:
DEFINE                DELETE
DISASSOCIATE
AIM> DEL ?
One of the following tokens expected:
IMAGE                 WINDOW
OBJECT_TYPE
AIM> DEL I ?
One of the following tokens expected:
IMAGE_NAME            IDENTIFIER
AIM> DEL I IM1
AIM> EXIT
-)




________________________________BOTTOM-OF-SCREEN____________________________
```

Comments: Exit the AIM.

Command: None.

4.2 TEXTUAL COMMANDS

4.2.1 Introduction

The AIM command language was designed to provide the user with a robust command set that is flexible and easy to use. These goals have been accomplished by allowing the user to specify a textual command in one of three different forms:

1. a full blown Ada-like syntax
2. a more terse abbreviated syntax
3. any mixture of the above two formats

For example, to create a new image called IM_1 the user has various choices:

1. CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IM_1
2. CREATE IMAGE IMAGE_NAME => IM_1
3. CR IMAGE IMAGE_NAME => IM_1
4. CREATE OBJECT_TYPE => IMAGE IM_1
5. CR OBJECT_TYPE => IMAGE IM_1
6. CR I IM_1

4.2.1.1 Command Parameters

The majority of the AIM textual commands have at least one formal parameter. The AIM supports only positional parameter association for its commands. The actual command parameters must be specified in the exact order of their formal parameter counterparts; however, the specification of the formal parameter names is optional.

In the spirit of overloading procedures in Ada, where applicable, AIM textual commands are overloaded by their first parameter. Typically a command's first parameter represents an object in the AIM environment. For example, the CREATE command can be used to create one of two AIM environmental objects, either a window or an image.

Note: For clarity and ease of reference, every AIM command that has an overloaded first parameter will be presented separately according to the alternate values of the command's first parameter. For instance, the CREATE command is qualified and explained as the CREATE IMAGE and CREATE WINDOW commands, where "CREATE IMAGE command" means the AIM CREATE command with "IMAGE" as its first parameter.

4.2.1.2 Command Description Format

Section 4.2 presents a thorough description of all the textual commands supported by the AIM. Where applicable, the following information is provided for each AIM textual command:

1. Command Name

2. Functional Description

3. Command Syntax

4. Command Parameters

5. Examples

6. Errors

Notation: The asterisk ("*") in the command format description denotes that all characters after the "*" are optional. Angle brackets ("< >") enclose required formal parameters.

4.2.2 ABORT_AIM

4.2.2.1 Functional Description

The ABORT_AIM command terminates all processes running under the AIM and exits from the AIM program. All active subordinate APSE processes are terminated without warning.

4.2.2.2 Syntax

```
AB*ORT
```

4.2.2.3 Command Parameters

None.

4.2.2.4 Examples

4.2.2.4.1 Long Form

```
AIM> ABORT_AIM
```

Terminate all processes running under the AIM and exit from the AIM program.

4.2.2.4.2 Mixed

Not applicable.

4.2.2.4.3 Short Form

```
AIM> AB
```

Terminate all processes running under the AIM and exit from the AIM program.

4.2.2.5 Errors

The semantic errors associated with this command include:

None

4.2.3 ASSOCIATE

4.2.3.1 Functional Description

The ASSOCIATE command maps a specified portion of a window onto an image. Assuming that n = <length> and p = <position> the ASSOCIATE command maps the last n lines of the specified window onto the given image starting at the pth line of the image. For example,

```
ASSOC WIN_1 IM_1 13 12
```

will map the last 12 lines of WIN_1 onto lines 13..24 of IM_1. This forms an abstract association between WIN_1 and IM_1 by creating a viewport (lines 13 thru 24) onto which the last 12 lines of WIN_1 are mapped. This viewport partitions the image vertically, and its width is equal to the width of the image.

Note: A window may be associated with more than one image at the same time; however, a specific window may only be associated with a specific image once.

4.2.3.2 Syntax

```
AS*SCCIATE  WINDOW_NAME  => <window name>
             IMAGE_NAME  => <image name>
             TOP         => <position>
             LENGTH      => <length>
```

4.2.3.3 Command Parameters

1. WINDOW_NAME => <window name>

   Specifies the window whose last <length> lines are to be mapped onto the specified image.

2. IMAGE_NAME => <image name>

   Specifies the image onto which a portion of the given window will be mapped.

3. TOP => <position>

   Specifies the starting position for the association relative to the top of the specified image.

4. LENGTH => <length>

   Specifies the length of the viewport used for the requested association. In addition to the viewport header, at least one line of a window must be displayed in a viewport; therefore, the minimum length allowable is 2.

4.2.3.4 Examples

4.2.3.4.1 Long Form

```
AIM> ASSOCIATE WINDOW_NAME=>WIN_1 IMAGE_NAME=>IM_1 TOP=>1 LENGTH=>24
```

Associate the entire contents of window WIN_1 with the image IM_1.

4.2.3.4.2 Mixed

```
AIM> ASSOC WIN_2 IMAGE_NAME => IM_2 1 LENGTH => 8
```

Associate the last 8 lines of window WIN_2 with the first 8 lines of image IM_1.

4.2.3.4.3 Short Form

```
AIM> AS WIN_3 IM_3 9 8
```

Associate the last 8 lines of window WIN_3 with lines 9 thru 16 of image IM_1.

4.2.3.5 Errors

The semantic errors associated with this command include:

1. "Window <window name> does not exist"

   The specified window name, <window name>, does NOT exist (i.e.--no window of that name has been created)

2. "Image <image name> does not exist"

   The specified image name, <image name>, does NOT exist (i.e.--no image of that name has been created)

3. "Invalid length": <length>

   The specified viewport length was either out of range, non-numeric, or too long to ensure non-intersecting viewports of the given image

4. "Invalid top line": <top_line>

   The specified top line for starting the viewport was either out of range, or non-numeric

5. "Association between specified window and image already exists"

   The specified window is currently associated with <image name> and the AIM prohibits multiple associations between the same window and image

4.2.4 CREATE IMAGE

4.2.4.1 Functional Description

The CREATE IMAGE command allows the AIM user to create a new image in the AIM environment. A newly created image will be alphabetically entered into the existing image list.

Note: There is no explicit limit to the number of images that can be created during an AIM session.

4.2.4.2 Syntax

```
CR*EATE  OBJECT_TYPE => I*MAGE,
          IMAGE_NAME  => <image name>
```

4.2.4.3 Command Parameters

1. OBJECT_TYPE => I*MAGE

   Specifies that a new image is to be created in the AIM environment.

2. IMAGE_NAME => <image name>

   Specifies the name of a new image to be created.

4.2.4.4 Examples

4.2.4.4.1 Long Form

```
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IM_1
```

Create an AIM image named IM_1.

4.2.4.4.2 Mixed

```
AIM> CREATE IMAGE IMAGE_NAME => IM_2
```

Create an AIM image named IM_2.

4.2.4.4.3 Short Form

```
AIM> CR I IM_3
```

Create an AIM image named IM_3.

4.2.4.5 Errors

The semantic errors associated with this command include:

1. "Image name, <image_name>, already in use"

   The specified image name already exists in the AIM environment

2. "Identifier name is too long"

   The specified image name was longer than 20 characters

4.2.5 CREATE WINDOW

4.2.5.1 Functional Description

The CREATE WINDOW command allows the AIM user to create a new window in the AIM environment. A newly created image will be alphabetically entered into the existing image list.

Note: There is no explicit limit to the number of windows that can be created during an AIM session.

The default values for the new window's flag settings follow:

1. SUSPENDS_OUTPUT_ON_FULL - true,

2. INPUT_COMPONENT_ACTIVE - false,

3. OUTPUT_COMPONENT_ACTIVE - false,

4. OUTPUT_SUSPENDED - false.

4.2.5.2 Syntax

```
  CR*EATE  OBJECT_TYPE => W*INDOW,
            WINDOW_NAME => <window name>
```

4.2.5.3 Command Parameters

1. OBJECT_TYPE => W*INDOW

   Specifies that a new window is to be created in the AIM environment.

2. WINDOW_NAME => <window name>

   Specifies the name of a new window to be created.

4.2.5.4 Examples

4.2.5.4.1 Long Form

```
  AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WIN_1
```

Create an AIM window named WIN_1.

4.2.5.4.2 Mixed

```
AIM> CREATE WINDOW WINDOW_NAME => WIN_2
```

Create an AIM window named WIN_2.

4.2.5.4.3 Short Form

```
AIM> CR W WIN_3
```

Create an AIM window named WIN_3.

4.2.5.5 Errors

The semantic errors associated with this command include:

1. "Window name, <window_name>, already in use"

   The specified window name already exists in the AIM environment

2. "Identifier name is too long"

   The specified window name was longer than 20 characters

4.2.6 DEFINE BINDING

4.2.6.1 Functional Description

The DEFINE BINDING command allows the AIM user to bind a new key sequence to the specified keystroke command. Initially the AIM assigns default values for the key sequences that are associated with the keystroke commands (see Appendix A). These bindings are maintained by the AIM CLI via an internal table. The DEFINE BINDING command redefines the key sequence bound to a specified keystroke command by instructing the AIM CLI to update the proper entry in the keystroke bindings table.

Note: The bindings established via the DEFINE BINDING command are in effect only for the lifetime of the AIM's execution; however, the DEFINE BINDING command may be included in the AIM's initialization script file to ensure that particular keystroke bindings exist every time the user enters an AIM session. The user may redefine the actual key sequence by altering the Terminal Capabilities File.

The valid AIM keystroke command names follow:

1. ABORT_SCRIPT
2. CLEAR_WINDOW
3. NEXT_IMAGE
4. NEXT_VIEWPORT
5. NEXT_PAGE
6. PREVIOUS_IMAGE
7. PREVIOUS_VIEWPORT
8. REDISPLAY_SCREEN
9. RETURN_TO_PREVIOUS_IMAGE
10. SWITCH_TO_AIM

4.2.6.2 Syntax

```
DEF*INE  OBJECT_TYPE  => B*INDING,
          KEY_NAME     => <keystroke command name>
          FUNCTION_KEY => <function_key>
```

4.2.6.3 Command Parameters

1. OBJECT_TYPE => B*INDING

   Specifies that a new keystroke binding is to be defined.

2. KEY_NAME => <keystroke command name>

   Specifies the name of the AIM keystroke command whose associated key sequence is to be altered.

3. FUNCTION_KEY => <function_key>

   Specifies the new key sequence to be bound to the specified AIM keystroke command.

4.2.6.4 Examples

4.2.6.4.1 Long Form

```
AIM> DEFINE OBJECT_TYPE=>BINDING KEY_NAME=>SWITCH_TO_AIM
   FUNCTION_KEY=> F10
```

Bind the function key F10 to the SWITCH_TO_AIM keystroke command.

4.2.6.4.2 Mixed

```
AIM> DEFINE BINDING KEY_NAME => CLEAR_WINDOW  F2
```

Bind the function key F2 to the CLEAR_WINDOW keystroke command.

4.2.6.4.3 Short Form

```
AIM> DEF B NEXT_VIEWPORT  F5
```

Bind the function key F5 to the NEXT_VIEWPORT keystroke command.

4.2.6.5 Errors

The semantic errors associated with this command include:

1. "Keyword: <keyword> can NOT be completed"

   The specified keystroke command name is invalid

4.2.7 DEFINE TERMINAL

4.2.7.1 Functional Description

The DEFINE TERMINAL command allows the AIM user to dynamically specify the type of physical terminal being used to communicate with the AIM. This command is used to ensure that the AIM's virtual terminal interprets and transmits the physical terminal's control sequences correctly.

4.2.7.2 Syntax

```
DEF*INE  OBJECT_TYPE   => T*ERMINAL,
          TERMINAL_NAME => <terminal type>
```

4.2.7.3 Command Parameters

1. OBJECT_TYPE => T*ERMINAL

   Specifies that the terminal type is to be changed.

2. TERMINAL_NAME => <terminal type>

   Specifies the new physical terminal identification.

4.2.7.4 Examples

4.2.7.4.1 Long Form

```
AIM> DEFINE OBJECT_TYPE => TERMINAL TERMINAL_NAME => vt100
```

Reset the user's terminal type to be a VT100.

Note: The AIM is capable of supporting those terminals listed in the Terminal Capabilities File which are directly supported by the underlying KAPSE (see Chapter 5 for a description of the Terminal Capabilities File).

4.2.7.4.2 Mixed

```
AIM> DEFINE OBJECT_TYPE => TERMINAL ibm3278
```

Reset the user's terminal type to be an IBM 3278.

4.2.7.4.3 Short Form

```
AIM> DEF T ti940
```

Reset the user's terminal type to be a TI940.

4.2.7.5 Errors

The semantic errors associated with this command include:

1. "Keyword: <keyword> can NOT be completed"

   The specified terminal name does not match any of the physical terminal names in the Terminal Capabilities File (see Chapter 5 for a description of how to create a Terminal Capabilities File entry)

4.2.8 DELETE IMAGE

4.2.8.1 Functional Description

The DELETE IMAGE command allows the user to delete an image currently active in the AIM environment. The image given as the input parameter is deleted from the internal list of all images defined in the AIM environment.

4.2.8.2 Syntax

```
DEL*ETE  OBJECT_TYPE => I*MAGE,
          IMAGE_NAME  => <image name>
```

4.2.8.3 Command Parameters

1. OBJECT_TYPE => I*MAGE

   Specifies that an image is to be deleted from the AIM environment.

2. IMAGE_NAME => <image name>

   Specifies the name of the image to be deleted.

4.2.8.4 Examples

4.2.8.4.1 Long Form

```
AIM> DELETE OBJECT_TYPE => IMAGE IMAGE_NAME => IM_1
```

Delete the image named IM_1.

4.2.8.4.2 Mixed

```
AIM> DELETE IMAGE IMAGE_NAME => IM_2
```

Delete the image named IM_2.

4.2.8.4.3 Short Form

```
AIM> DEL I IM_3
```

Delete the image named IM_3.

4.2.8.5 Errors

The semantic errors associated with this command include:

1. "Image <image name> does not exist"

   The specified image does not exist in the AIM environment

2. "Deleting AIM image prohibited"

   The specified image was the AIM image and, it can NOT be deleted

4.2.9 DELETE WINDOW

4.2.9.1 Functional Description

The DELETE WINDOW command allows the user to delete a window currently active in the AIM environment. The window given as the input parameter is deleted from the internal list of all windows defined in the AIM environment. If the specified window has an active pad, the AIM will prompt the user as to the disposition of the corresponding pad file(s).

4.2.9.2 Syntax

```
DEL*ETE  OBJECT_TYPE   => W*INDOW,
          WINDOW_NAME   => <window name>
```

4.2.9.3 Command Parameters

1. OBJECT_TYPE => W*INDOW

   Specifies that a window is to be deleted from the AIM environment.

2. WINDOW_NAME => <window name>

   Specifies the name of the window to be deleted.

4.2.9.4 Examples

4.2.9.4.1 Long Form

```
AIM> DELETE OBJECT_TYPE => WINDOW WINDOW_NAME => WIN_1
```

Delete the window named WIN_1.

4.2.9.4.2 Mixed

```
AIM> DELETE WINDOW WINDOW_NAME => WIN_2
```

Delete the window named WIN_2.

4.2.9.4.3 Short Form

```
AIM> DEL W WIN_3
```

Delete the window named WIN_3.

4.2.9.5 Errors

The semantic errors associated with this command include:

1. "Window <window name> does not exist"

   The specified window does not exist in the AIM environment

2. "Deleting AIM window prohibited"

   The specified window was the AIM window and it CANNOT be deleted

3. "Subordinate Programs Exist."

   Cannot delete a window with subordinate programs running

4.2.10 DISASSOCIATE

4.2.10.1 Functional Description

The DISASSOCIATE command removes the association between the specified window and image. The appropriate window-image association is deleted from the AIM's internal viewport list.

4.2.10.2 Syntax

```
  DI*SASSOCIATE  WINDOW_NAME => <window name>
                  IMAGE_NAME  => <image name>
```

4.2.10.3 Command Parameters

1. WINDOW_NAME => <window name>

   Specifies the window to be disassociated from the specified image.

2. IMAGE_NAME => <image name>

   Specifies the image from which the specified viewport will be deleted.

4.2.10.4 Examples

4.2.10.4.1 Long Form

```
AIM> DISASSOCIATE WINDOW_NAME => WIN_1 IMAGE_NAME => IM_1
```

Remove the association between WIN_1 and IM_1.

4.2.10.4.2 Mixed

```
AIM> DISASSOCIATE WINDOW_NAME => WIN_2 IM_2
```

Remove the association between WIN_2 and IM_2.

4.2.10.4.3 Short Form

```
AIM> DI WIN_3 IM_3
```

Remove the association between WIN_3 and IM_3.

4.2.10.5 Errors

The semantic errors associated with this command include:

1. "Window <window name> does not exist"

   The specified window name, <window name>, does NOT exist (i.e.--no window of that name has been created)

2. "Image <image name> does not exist"

   The specified image name, <image name>, does NOT exist (i.e.--no image of that name has been created)

3. "Cannot disassociate the AIM window from the AIM image"

   An attempt to disassociate the AIM window and image was made; the AIM program prohibits the user from disassociating the AIM window and image

4.2.11 EXIT

4.2.11.1 Functional Description

The EXIT command will terminate execution of the AIM only if ALL subordinate programs have terminated, unlike the ABORT command which will terminate the execution of the AIM regardless of the current AIM environment.

4.2.11.2 Syntax

```
E*XIT
```

4.2.11.3 Command Parameters

None.

4.2.11.4 Examples

4.2.11.4.1 Long Form

```
AIM> EXIT
```

Gracefully exit the AIM program.

4.2.11.4.2 Mixed

Not applicable.

4.2.11.4.3 Short Form

```
AIM> E
```

Gracefully exit the AIM program.

4.2.11.5 Errors

The semantic errors associated with this command include:

1. "Subordinate Programs Exist."

   The AIM CANNOT be left with subordinate programs running.

4.2.12 GOTO IMAGE

4.2.12.1 Functional Description

The GOTO IMAGE command moves the cursor to the top of the named image. The specified image is then logically connected to the terminal's display.

4.2.12.2 Syntax

```
G*OTO  OBJECT_TYPE => I*MAGE,
         IMAGE_NAME  => <image name>
```

4.2.12.3 Command Parameters

1. OBJECT_TYPE => I*MAGE

   Specifies that the cursor will be positioned in the top viewport of an image.

2. IMAGE_NAME => <image name>

   Specifies the image to logically connect to the terminal's display.

4.2.12.4 Examples

4.2.12.4.1 Long Form

```
AIM> GOTO OBJECT_TYPE => IMAGE IMAGE_NAME => IM_1
```

Position the cursor at the top of IM_1.

4.2.12.4.2 Mixed

```
AIM> GOTO IMAGE IMAGE_NAME => IM_2
```

Position the cursor at the top of IM_2.

4.2.12.4.3 Short Form

```
AIM> G I IM_3
```

Position the cursor at the top of IM_3.

4.2.12.5 Errors

The semantic errors associated with this command include:

1. "Image <image name> does not exist"

   The specified image name, <image name>, does NOT exist (i.e.--no image of that name has been created)

2. "Image, <image_name>, not associated with a window"

   The specified image name, <image name>, is NOT associated with a window

4.2.13 GOTO WINDOW

4.2.13.1 Functional Description

The GOTO WINDOW command moves the cursor to the named window. If the specified window is mapped onto more than one image, the cursor is positioned at the top of the largest viewport associated with that window. Conflicts between viewports of identical size shall be resolved alphabetically by image names.

4.2.13.2 Syntax

```
G*OTO  OBJECT_TYPE => W*INDOW,
        WINDOW_NAME => <window name>
```

4.2.13.3 Command Parameters

1. OBJECT_TYPE => W*INDOW

   Specifies that the cursor will be positioned at the top of a window.

2. WINDOW_NAME => <window name>

   Specifies the window name that is to become the current window on the user's screen.

4.2.13.4 Examples

4.2.13.4.1 Long Form

```
AIM> GOTO OBJECT_TYPE => WINDOW WINDOW_NAME => WIN_1
```

Position the cursor in the largest viewport associated with WIN_1.

4.2.13.4.2 Mixed

```
AIM> GOTO OBJECT_TYPE => WINDOW  WIN_2
```

Position the cursor in the largest viewport associated with WIN_2.

4.2.13.4.3 Short Form

```
AIM> G W WIN_3
```

Position the cursor in the largest viewport associated with

WIN_3.

4.2.13.5 Errors

The semantic errors associated with this command include:

1. "Window <window name> does not exist"

   The specified window name, <window name>, does NOT exist (i.e.--no window of that name has been created)

2. "<window_name> not associated with any images"

   The specified window name, <window name>, is NOT associated with an image

4.2.14 HELP

4.2.14.1 Functional Description

The HELP command invokes the AIM HELP utility, which provides information to the user about the operation of the AIM commands.

4.2.14.2 Syntax

```
  H*ELP  ( <identifier> {<identifier>} {*}  )
```

4.2.14.3 Command Parameters

1. <identifier> {<identifier>} {*})

   Specifies one or more AIM keywords that indicate what information is to be displayed. If you specify an astrisk (*) after any keyword, ALL help information available in the AIM HELP file at that level is displayed.

4.2.14.4 Examples

4.2.14.4.1 Long Form

```
AIM> HELP ASSOCIATE PARAMETERS
```

Display all the available AIM help information describing the parameters of the ASSOCIATE command.

4.2.14.4.2 Mixed

```
AIM> HELP
```

Display a list of the available top level AIM HELP topics.

4.2.14.4.3 Short Form

```
AIM> H
```

Display a list of the available top level AIM HELP topics.

4.2.14.5 Errors

The semantic errors associated with this command include:

1. "Sorry, no documentation available on <subject>"

   There is no AIM help information in the HELP file available for the specified keyword (<subject>)

4.2.15 INFO

4.2.15.1 Functional Description

The INFO command invokes the AIM INFO utility which provides the user with AIM environmental information such as information about images, windows, keystroke command associations, and the terminal type.

4.2.15.2 Syntax

```
I*NFO  ( <identifier> {<identifier>} {*}  )
```

4.2.15.3 Command Parameters

1. <identifier> {<identifier>} {*})

   Specifies one or more AIM objects that indicate what information is to be displayed. If you specify an astrisk (*) after any object, ALL AIM environmental information available for that object is displayed.

4.2.15.4 Examples

4.2.15.4.1 Long Form

```
AIM> INFO WINDOWS WIN_1 MODES
```

Display the window flag settings for WIN_1.

4.2.15.4.2 Mixed

```
AIM> INFO WINDOWS
```

Display an alphabetical list of ALL the current windows in the AIM environment.

4.2.15.4.3 Short Form

```
AIM> I
```

Display a list of ALL objects for which information is available.

4.2.15.5 Errors

The semantic errors associated with this command include:

1. "Invalid identifier: <identifier>"

   The current AIM INFO command was invalid

2. "Unrecognized INFO keyword word, <keyword>"

   There is no AIM information available for the specified AIM keyword

3. "Invalid INFO command after: <input>"

   The user input was not valid after the specified token.

4. "Invalid INFO command beginning at: <input>"

   The user input was not valid beginning at the specified token.

5. "Image <image_name> does not exist"

   The specified image does not exist

6. "Window <window_name> does not exist"

   The specified window does not exist

4.2.16 RESET AIM_SUSPENDS

4.2.16.1 Functional Description

The RESET AIM_SUSPENDS command assigns a boolean value of FALSE to the AIM global variable, SUSPEND_ON_AIM. When this flag is off, no APSE program output suspension occurs when the SWITCH_TO_AIM keystroke is depressed. The value of this variable is checked by the CI each time the SWITCH_TO_AIM keystroke is depressed by the user. If it is TRUE, the CI suspends the output of every APSE program executing underneath the AIM, otherwise, no special action is taken. The default value for this flag is TRUE.

4.2.16.2 Syntax

```
RESE*T MODE_TYPE   => A*IM_SUSPENDS )
```

4.2.16.3 Command Parameters

1. MODE_TYPE => A*IM_SUSPENDS

   Reset the SUSPEND_ON_AIM flag.

4.2.16.4 Examples

4.2.16.4.1 Long Form

```
AIM> RESET MODE_TYPE => AIM_SUSPENDS
```

Reset the global SUSPEND_ON_AIM flag to FALSE.

4.2.16.4.2 Mixed

```
AIM> RESET AIM_SUSPENDS
```

Reset the global SUSPEND_ON_AIM flag to FALSE.

4.2.16.4.3 Short Form

```
AIM> RESE A
```

Reset the global SUSPEND_ON_AIM flag to FALSE.

4.2.16.5 Errors

The semantic errors associated with this command include:

None

4.2.17 RESET FULL

4.2.17.1 Functional Description

The RESET FULL command assigns a boolean value of FALSE to the specified window's SUSPENDS_OUTPUT_WHEN_FULL flag. When this flag setting is turned off for a window, the window's output will scroll continuously. The default setting for every window is TRUE.

4.2.17.2 Syntax

```
RESE*T  MODE_TYPE   => F*ULL,
         WINDOW_NAME => <window name>
```

4.2.17.3 Command Parameters

1. MODE_TYPE => F*ULL

   Reset the SUSPEND_OUTPUT_ON_FULL flag for the specified window.

2. WINDOW_NAME => <window name>

   Specifies the window for which the SUSPEND_WHEN_FULL flag is to be turned off.

4.2.17.4 Examples

4.2.17.4.1 Long Form

```
AIM> RESET MODE_TYPE => FULL WINDOW_NAME => WIN_1
```

Turn off the SUSPEND_WHEN_FULL flag associated with WIN_1. This will allow continuous scrolling of WIN_1's output.

4.2.17.4.2 Mixed

```
AIM> RESET FULL WINDOW_NAME => WIN_2
```

Turn off the SUSPEND_WHEN_FULL flag associated with WIN_2. This will allow continuous scrolling of WIN_2's output.

4.2.17.4.3 Short Form

```
AIM> RESE F WIN_3
```

Turn off the SUSPEND_WHEN_FULL flag associated with WIN_3. This will allow continuous scrolling of WIN_3's output.

4.2.17.5 Errors

The semantic errors associated with this command include:

1. "Window <window name> does not exist"

   The specified window name, <window name>, does NOT exist (i.e.--no window of that name has been created)

4.2.18 RESET INPUT_PAD

4.2.18.1 Functional Description

The RESET INPUT_PAD command deactivates the input pad associated with the specified window, and thus, closes the corresponding database file.

4.2.18.2 Syntax

```
  RESE*T  MODE_TYPE    => I*NPUT_PAD,
          WINDOW_NAME => <window name>
```

4.2.18.3 Additional Parameters

1. MODE_TYPE => I*INPUT_PAD

   Reset the INPUT_COMPONENT_ACTIVE flag for the specified window.

2. WINDOW_NAME => <window name>

   Specifies the window for which the input pad is to be deactivated.

4.2.18.4 Examples

4.2.18.4.1 Long Form

```
AIM> RESET MODE_TYPE => INPUT_PAD WINDOW_NAME => WIN_1
```

Deactivate the input pad for WIN_1. This causes the corresponding database file to be closed.

4.2.18.4.2 Mixed

```
AIM> RESET INPUT_PAD WINDOW_NAME => WIN_2
```

Deactivate the input pad for WIN_2. This causes the corresponding database file to be closed.

4.2.18.4.3 Short Form

```
AIM> RESE I WIN_3
```

Deactivate the input pad for WIN_3. This causes the corresponding database file to be closed.

4.2.18.5 Errors

The semantic errors associated with this command include:

1. "Window <window name> does not exist"

   The specified window name, <window name>, does NOT exist (i.e.--no window of that name has been created)

4.2.19 RESET OUTPUT_PAD

4.2.19.1 Functional Description

The RESET OUTPUT_PAD command deactivates the output pad associated with the specified window, and thus, closes the corresponding database file.

4.2.19.2 Syntax

```
  RESE*T  MODE_TYPE   => O*UTPUT_PAD,
           WINDOW_NAME => <window name>
```

4.2.19.3 Command Parameters

1. MODE_TYPE => O*UTPUT_PAD

   Reset the OUTPUT_COMPONENT_ACTIVE flag for the specified window.

2. WINDOW_NAME => <window name>

   Specifies the window for which the output pad is to be deactivated.

4.2.19.4 Examples

4.2.19.4.1 Long Form

```
AIM> RESET MODE_TYPE => OUTPUT_PAD WINDOW_NAME => WIN_1
```

Deactivate the output pad for WIN_1, causing the corresponding database file to be closed.

4.2.19.4.2 Mixed

```
AIM> RESET OUTPUT_PAD WINDOW_NAME => WIN_2
```

Deactivate the output pad for WIN_2.

4.2.19.4.3 Short Form

```
AIM> RESE I WIN_3
```

Deactivate the output pad for WIN_3.

4.2.19.5 Errors

The semantic errors associated with this command include:

1. "Window <window name> does not exist"

   The specified window name, <window name>, does NOT exist (i.e.--no window of that name has been created)

4.2.20 RESET PADS

4.2.20.1 Functional Description

The RESET PADS command deactivates both the input and output pad file associated with the specified window, and thus, closes the corresponding database files.

4.2.20.2 Syntax

```
  RESE*T  MODE_TYPE    => P*ADS,
           WINDOW_NAME => <window name>
```

4.2.20.3 Command Parameters

1. MODE_TYPE => P*ADS

   Reset the OUTPUT_COMPONENT_ACTIVE and INPUT_COMPONENT_ACTIVE flags for the specified window.

2. WINDOW_NAME => <window name>

   Specifies the window for which the input and output pads are to be deactivated.

4.2.20.4 Examples

4.2.20.4.1 Long Form

```
  AIM> RESET MODE_TYPE=>PADS WINDOW_NAME=>WIN_1
```

Deactivate the input and output pads for WIN_1.

4.2.20.4.2 Mixed

```
  AIM> RESET PADS WIN_2
```

Deactivate the input and output pads for WIN_2.

4.2.20.4.3 Short Form

```
  AIM> RESE P WIN_3
```

Deactivate the input and output pads for WIN_3.

4.2.20.5 Errors

The semantic errors associated with this command include:

1. "Window <window name> does not exist"

   The specified window name, <window name>, does NOT exist (i.e.--no window of that name has been created)

AD-A198 151

NAVAL OCEAN SYSTEMS CENTER, SAN DIEGO, CA
APSE INTERACTIVE MONITOR – USER'S MANUAL-ADE™
VERSION BY: TEXAS INSTRUMENTS INC.

2 OF 4
NOSC TD 121
UNCLASSIFI
JAN 1988

UNCLASSIFIED U

4.2.21 RESUME EXECUTION

4.2.21.1 Functional Description

The RESUME EXECUTION command directs the AIM to resume execution of the specified window's suspended APSE program. This information is displayed in the program status field of the window's viewport header ("Running"). If the specified window's APSE program is not suspended, nothing happens.

4.2.21.2 Syntax

```
RESU*ME  OBJECT_TYPE => E*XECUTION,
          WINDOW_NAME => <window name>
```

4.2.21.3 Command Parameters

1. OBJECT_TYPE => E*XECUTION

   Specifies that the execution of the APSE program associated with the specified window is to be resumed.

2. WINDOW_NAME => <window name>

   Specifies the window whose associated APSE program execution is to be resumed.

4.2.21.4 Examples

4.2.21.4.1 Long Form

```
AIM> RESUME OBJECT_TYPE => EXECUTION WINDOW_NAME => WIN_1
```

Resume the execution of the APSE program associated with WIN_1.

4.2.21.4.2 Mixed

```
AIM> RESUME EXECUTION WINDOW_NAME => WIN_2
```

Resume the execution of the APSE program associated with W[illegible]2.

4.2.21.4.3 Short Form

```
AIM> RESU E WIN_3
```

Resume the execution of the APSE program associated with WIN_3.

4.2.21.5 Errors

The semantic errors associated with this command include:

1. "Window <window name> does not exist"

   The specified window name, <window name> does NOT exist (i.e.--no window of that name has been created)

4.2.22 RESUME PROGRAM OUTPUT [illegible]

4.2.22.1 Functional [illegible]

The RESUME PROGRAM [illegible] AIM to resume the output of the specified window [illegible] the output from the APSE program associated [illegible] is not suspended, nothing happens; otherwise the "S [illegible] viewport header is turned off.

4.2.22.2 Syntax

```
RESU*ME  OBJECT_TYPE => P*ROGRAM_OUTPUT,
          WINDOW_NAME => <window name>
```

4.2.22.3 Command Parameters

1. OBJECT_TYPE => W*INDOW_OUTPUT

   Specifies that the output of the APSE program associated with the specified window is to be resumed.

2. WINDOW_NAME => <window name>

   Specifies the window whose APSE program's output is to be resumed.

4.2.22.4 Examples

4.2.22.4.1 Long Form

```
AIM> RESUME OBJECT_TYPE => PROGRAM_OUTPUT WINDOW_NAME => WIN_1
```

Resume the output of the APSE program associated with WIN_1.

4.2.22.4.2 Mixed

```
AIM> RESUME PROGRAM_OUTPUT WINDOW_NAME => WIN_2
```

Resume the output of the APSE program associated with WIN_2.

4.2.22.4.3 Short Form

```
AIM> RESU E WIN_3
```

Resume the output of the APSE program associated with WIN_3.

4.2.22.5 Errors

The semantic errors associated with this command include:

1. "Window <window name> does not exist"

   The specified window name, <window name> does NOT exist (i.e.--no window of that name has been created)

4.2.23 SCRIPT

4.2.23.1 Functional Description

In addition to the normal method of entering AIM commands interactively through the terminal, a user may compose command scripts to be read and executed by the AIM. A command script is simply a KAPSE database file which contains AIM commands. The user may create these command files using an editor available in an APSE, or AIM sessions may be logged into pads which may be edited and used for command scripts later. To execute a command script, the user enters the SCRIPT command and provides the name of the file containing the script. As a command script is executed, each command is echoed in the AIM window. The AIM provides an "ABORT SCRIPT" keystroke command to terminate the execution of a command script.

Upon invocation, the AIM looks for a user-created Initialization Script file. The name of this script file is specified by the user when the AIM program is invoked. If the file exists, the AIM automatically performs an implicit Switch-to-AIM and executes the command script. If the file does not exist, no error message is generated and the AIM session will begin as usual. The Initialization Script benefits the user with a specific application, who wishes to set up a series of "predefined" windows and images.

Though the concept of the command script is simple, there are some subtle implications associated with its use, which are explained below.

1. GOTO command in a script

   A GOTO command in a command script switches the current image being displayed to something other than the AIM image. Therefore, if the commands in the command script are sent to the new image, they will not be received by the AIM command interpreter. The commands will not be recognized and will cause errors. To alleviate this problem, the AIM stacks input sources and switches from script execution to interactive mode upon encountering a GOTO command. The user may then interact with the AIM through the current window. When the user presses the SWITCH_TO_AIM key sequence, the execution of the previous command script will resume. The user may allow the command script to run to completion, or may abort the script with a keystroke command.

2. Keystroke commands in a script

   Since none of the keystroke commands are applicable during the execution of a command script, (except, of course, ABORT_SCRIPT, which would not make much sense inside a script file), keystroke commands are not allowed in command scripts.

3. Error recovery from a script

   AIM textual commands in a script file are processed exactly as if the user had entered them from the terminal. However, when the AIM detects an error in a command script, the AIM sends an error message to the terminal and terminates the command script. This is to ensure that errors do not propagate through a command script (such as an error in a CREATE command which could affect subsequent ASSOCIATE commands) and possibly jeopardize user data or files.

4. Recursive or mutually recursive command scripts

   Command scripts may invoke other command scripts by the use of the SCRIPT command. When the new command script ends, control is returned to the "calling" command script, and execution of that command script is resumed. The AIM command language contains no conditional statements, which means that it would be impossible to end recursion once it was started. For this reason, a command script that calls itself or two command scripts which call each other would create an infinite loop. The AIM cannot explicitly detect recursive scripts; however, recursive execution will be prohibited since a script(text) file may be opened only once at any given time [DOD83], so the user must take measures to ensure recursion does not occur. The AIM imposes a maximum script nesting level of seven; exceeding this limit will cause an error message to be generated and the execution of all command scripts to abort.

4.2.23.2 Syntax

```
SC*RIPT  FILE_NAME => <file name>
```

4.2.23.3 Command Parameters

1. FILE_NAME => <file name>

   Specifies the file which contains the script.

4.2.23.4 Examples

4.2.23.4.1 Long Form

```
AIM> SCRIPT FILE_NAME => script1
```

Execute the script with file name "script1" (File specifications are

APSE dependent).

4.2.23.4.2 Mixed

None.

4.2.23.4.3 Short Form

```
AIM> SC script1
```

Execute the script with file name "script1". (File specifications are APSE dependent.)

4.2.23.5 Errors

The semantic errors associated with this command include:

1. "No such script file: <file name>"

   The specified file name does NOT exist

2. "File cannot be opened"

   This error is caused when a script file cannot be opened; possible causes of this error include:

   1. the script file is already open

   2. the AIM does not have read access for the script file

3. "Maximum script nesting level surpassed"

   The maximum nesting level of scripts (7) was surpassed

4.2.24 SET AIM_SUSPENDS

4.2.24.1 Functional Description

The SET AIM_SUSPENDS command assigns a boolean value of TRUE to the AIM global variable, SUSPEND_ON_AIM. When this flag is on, APSE program output suspension occurs when the SWITCH_TO_AIM keystroke is depressed. The value of this variable is checked by the CI each time the SWITCH_TO_AIM keystroke is depressed by the user. If it is TRUE, the CI suspends the output of every APSE program executing underneath the AIM, otherwise, no action is taken. The default value for this flag is TRUE.

4.2.24.2 Syntax

```
  SE*T  MODE_TYPE => A*IM_SUSPENDS )
```

4.2.24.3 Command Parameters

1. MODE_TYPE => A*IM_SUSPENDS

   Set the SUSPEND_ON_AIM flag.

4.2.24.4 Examples

4.2.24.4.1 Long Form

```
  AIM> SET MODE_TYPE => AIM_SUSPENDS
```

Set the global SUSPEND_ON_AIM flag to TRUE.

4.2.24.4.2 Mixed

```
  AIM> SET AIM_SUSPENDS
```

Set the global SUSPEND_ON_AIM flag to TRUE.

4.2.24.4.3 Short Form

```
  AIM> SE A
```

Set the global SUSPEND_ON_AIM flag to TRUE.

4.2.24.5 Errors

The semantic errors associated with this command include:

None

4.2.25 SET FULL

4.2.25.1 Functional Description

The SET FULL command assigns a boolean value of TRUE to the specified window's SUSPENDS_OUTPUT_WHEN_FULL flag. When this flag setting is turned on for a window, the window's output will NOT scroll continuously, it will be suspended when a full window of output data has been generated. The default setting for every window is TRUE.

4.2.25.2 Syntax

```
SE*T  MODE_TYPE   => F*ULL,
       WINDOW_NAME => <window name>
```

4.2.25.3 Command Parameters

1. MODE_TYPE => F*ULL

   Set the SUSPEND_OUTPUT_ON_FULL flag for the specified window.

2. WINDOW_NAME => <window name>

   Specifies the window for which the SUSPEND_WHEN_FULL flag is to be turned on.

4.2.25.4 Examples

4.2.25.4.1 Long Form

```
AIM> SET MODE_TYPE => FULL WINDOW_NAME => WIN_1
```

Turn on the SUSPEND_WHEN_FULL flag associated with WIN_1. This will stop continuous scrolling of WIN_1's output.

4.2.25.4.2 Mixed

```
AIM> SET FULL WINDOW_NAME => WIN_2
```

Turn on the SUSPEND_WHEN_FULL flag associated with WIN_2. This will stop continuous scrolling of WIN_2's output.

4.2.25.4.3 Short Form

```
AIM> SE F WIN_3
```

Turn on the SUSPEND_WHEN_FULL flag associated with WIN_3. This will stop continuous scrolling of WIN_3's output.

4.2.25.5 Errors

The semantic errors associated with this command include:

1. "Window <window name> does not exist"

   The specified window name, <window name>, does NOT exist (i.e.--no window of that name has been created)

4.2.26 SET INPUT_PAD

4.2.26.1 Functional Description

The SET INPUT_PAD command activates the input pad associated with the specified window, and thus, opens a database file corresponding to the input component of the specified window's pad. When creating an input component for a window which has already had a pad opened for it (and then closed) a unique database file will be created for the new component.

When a window's input pad is active, all input destined for the specified window is also written (logged) to an input pad file. The AIM automatically generates a unique pad file name. The AIM command interpreter positively re-enforces the user by displaying a message indicating the name of the created file.

4.2.26.2 Syntax

```
  SE*T  MODE_TYPE    => I*NPUT_PAD,
         WINDOW_NAME => <window name>
```

4.2.26.3 Command Parameters

1. MODE_TYPE => I*INPUT_PAD

   Set the INPUT_COMPONENT_ACTIVE flag for the specified window.

2. WINDOW_NAME => <window name>

   Specifies the window for which the input pad is to be activated.

4.2.26.4 Examples

4.2.26.4.1 Long Form

```
  AIM> SET MODE_TYPE => INPUT_PAD WINDOW_NAME => WIN_1
  ***AIM generated input pad file name: WIN_1.INP
```

Activate the input pad for WIN_1.

Note: the pad file names generated by the AIM shall adhere to the underlying KAPSE's file naming conventions.

4.2.26.4.2 Mixed

```
AIM> SET INPUT_PAD WINDOW_NAME => WIN_2
```

Activate the input pad for WIN_2. The AIM generated the file name of WIN_2's input pad, namely WIN_2.INP.

Note: the pad file names generated by the AIM shall adhere to the underlying KAPSE's file naming conventions.

4.2.26.4.3 Short Form

```
AIM> SE I WIN_3
***AIM generated input pad file name: WIN_3.INP
```

Activate the input pad for WIN_3. The AIM generated the file name of WIN_3's input pad, namely WIN_3.INP.

Note: the pad file names generated by the AIM shall adhere to the underlying KAPSE's file naming conventions.

4.2.26.5 Errors

The semantic errors associated with this command include:

1. "Window <window name> does not exist"

   The specified window name, <window name>, does NOT exist (i.e.--no window of that name has been created)

2. "Input Pad for window <window_name> is already active"

   An input pad is already active for the specified window

3. "File cannot be opened"

   The specified pad file cannot be opened; possible causes of this error include:

   1. the file is already open

   2. the AIM does not have read access for the file

4.2.27 SET OUTPUT_PAD

4.2.27.1 Functional Description

The SET OUTPUT_PAD command activates the output pad associated with the specified window, and thus, opens the database file associated with the output component of the specified window's pad. When creating an output component for a window which has already had a pad opened for it (and then closed), a unique database file will be created for the new component.

When a window's output pad is active, all output destined for the specified window is also written (logged) to an output pad file. The AIM automatically generates a unique pad file name. The AIM command interpreter positively re-enforces the user by displaying a message indicating the name of the created file.

4.2.27.2 Syntax

```
  SE*T  MODE_TYPE    => O*UTPUT_PAD,
         WINDOW_NAME => <window name>
```

4.2.27.3 Command Parameters

1. MODE_TYPE => O*UTPUT_PAD

   Set the OUTPUT_COMPONENT_ACTIVE flag for the specified window.

2. WINDOW_NAME => <window name>

   Specifies the window for which the output pad is to be activated.

4.2.27.4 Examples

4.2.27.4.1 Long Form

```
  AIM> SET MODE_TYPE => OUTPUT_PAD WINDOW_NAME => WIN_1
  ***AIM generated output pad file name: WIN_1.OUT
```

Activate the output pad for WIN_1. The AIM CLI generated the file name of WIN_1's output pad, namely WIN_1.OUT.

Note: the pad file names generated by the AIM shall adhere to the underlying KAPSE's file naming conventions.

4.2.27.4.2 Mixed

```
AIM> SET OUTPUT_PAD WINDOW_NAME => WIN_2
***AIM generated output pad file name: WIN_2.OUT
```

Activate the output pad for WIN_2. The AIM CLI generated the file name of WIN_2's output pad, namely WIN_2.OUT.

Note: the pad file names generated by the AIM shall adhere to the underlying KAPSE's file naming conventions.

4.2.27.4.3 Short Form

```
AIM> SE O WIN_3
***AIM generated output pad file name: WIN_3.OUT
```

Activate the output pad for WIN_3. The AIM CLI generated the file name of WIN_3's output pad, namely WIN_3.OUT.

Note: the pad file names generated by the AIM shall adhere to the underlying KAPSE's file naming conventions.

4.2.27.5 Errors

The semantic errors associated with this command include:

1. "Window <window name> does not exist"

   The specified window name, <window name>, does NOT exist (i.e.--no window of that name has been created)

2. "Output Pad for window: <window_name> is already active"

   An output pad is already active for the specified window

3. "File cannot be opened"

   The specified pad file cannot be opened. Possible causes of this error include:

   1. the file is already open

   2. the AIM does not have read access for the file

4.2.28 SET PADS

4.2.28.1 Functional Description

The SET PADS command activates both an input and output pad file for the specified window, and thus, opens database files associated with the input and output component of the specified window's pad. When creating an input or output component for a window which has already had a pad opened for it (and then closed), a unique database file will be created for the new component(s).

When a window's input pad is active, all input destined for the specified window is also written (logged) to an input pad file, and when an output pad is active, all output destined for the specified window is also logged in the output pad file. The AIM automatically generates a unique pad file name. The AIM command interpreter positively re-enforces the user by displaying a message indicating the name of the created file.

4.2.28.2 Syntax

```
  SE*T  MODE_TYPE    => P*ADS,
         WINDOW_NAME => <window name>
```

4.2.28.3 Command Parameters

1. MODE_TYPE => P*ADS

   Set the OUTPUT_COMPONENT_ACTIVE and INPUT_COMPONENT_ACTIVE flags for the specified window.

2. WINDOW_NAME => <window name>

   Specifies the window for which the input and output pads are to activated.

4.2.28.4 Examples

4.2.28.4.1 Long Form

```
  AIM> SET MODE_TYPE => PADS WINDOW_NAME => WIN_1
```

Activate the input and output pads for WIN_1.
```
***AIM generated input pad file name:  WIN_1.INP
***AIM generated output pad file name:  WIN_1.OUT
```

The AIM command interpreter generated the input and output pad file names to be associated with WIN_1, namely WIN_1.INP and

WIN_1.OUT.

Note: the pad file names generated by the AIM shall adhere to the underlying KAPSE's file naming conventions.

4.2.28.4.2 Mixed

```
AIM> SET PADS WIN_2
```

Activate the input and output pads for WIN_2.
***AIM generated input pad file name: WIN_2.INP
***AIM generated output pad file name: WIN_2.OUT

The AIM command interpreter generated the input and output pad file names to be associated with WIN_2, namely WIN_2.INP and WIN_2.OUT.

Note: the pad file names generated by the AIM shall adhere to the underlying KAPSE's file naming conventions.

4.2.28.4.3 Short Form

```
AIM> SE P WIN_3
***AIM generated input pad file name:  WIN_3.INP
***AIM generated output pad file name:  WIN_3.OUT
```

Activate the input and output pads for WIN_3. The AIM command interpreter generated the input and output pad file names to be associated with WIN_3, namely WIN_3.INP and WIN_3.OUT.

Note: the pad file names generated by the AIM shall adhere to the underlying KAPSE's file naming conventions.

4.2.28.5 Errors

The semantic errors associated with this command include:

1. "Window <window name> does not exist"

   The specified window name, <window name>, does NOT exist (i.e.--no window of that name has been created)

2. "Input Pad for window: <window_name> is already active"

   An input pad is already active for the specified window

3. "Output Pad for window: <window_name> is already active"

   An output pad is already active for the specified window

4. "File cannot be opened"

The specified pad file cannot be opened; possible causes of this error include:

1. the file is already open

2. the AIM does not have read access for the file

4.2.29 SUSPEND EXECUTION

4.2.29.1 Functional Description

The SUSPEND EXECUTION command directs the AIM to suspend execution of the specified window's APSE program. This information is displayed in the program status field of the window's viewport header ("Suspended"). If the specified window's APSE program is already suspended, nothing happens.

4.2.29.2 Syntax

```
SU*SPEND  OBJECT_TYPE => E*XECUTION,
           WINDOW_NAME => <window name>
```

4.2.29.3 Command Parameters

1. OBJECT_TYPE => E*XECUTION

   Specifies that the execution of the APSE program associated with the specified window is to be suspended.

2. WINDOW_NAME => <window name>

   Specifies the window whose associated APSE program execution is to be suspended.

4.2.29.4 Examples

4.2.29.4.1 Long Form

```
AIM> SUSPEND OBJECT_TYPE => EXECUTION WINDOW_NAME => WIN_1
```

Suspend the execution of the APSE program associated with WIN_1.

4.2.29.4.2 Mixed

```
AIM> SUSPEND EXECUTION WINDOW_NAME => WIN_2
```

Suspend the execution of the APSE program associated with WIN_2.

4.2.29.4.3 Short Form

```
AIM> SU E WIN_3
```

Suspend the execution of the APSE program associated with WIN_3.

4.2.29.5 Errors

The semantic errors associated with this command include:

1. "Window <window name> does not exist"

   The specified window name, <window name>, does NOT exist (i.e.--no window of that name has been created)

4.2.30 SUSPEND PROGRAM_OUTPUT

4.2.30.1 Functional Description

The SUSPEND PROGRAM_OUTPUT command directs the AIM to suspend the output of the specified window's APSE program. If the output from the APSE program associated with the specified window is already suspended then nothing happens; otherwise, the 'S' flag in the viewport header is turned on.

4.2.30.2 Syntax

```
SU*SPEND  OBJECT_TYPE => W*INDOW_OUTPUT,
           WINDOW_NAME => <window name>
```

4.2.30.3 Command Parameters

1. OBJECT_TYPE => W*INDOW_OUTPUT

   Specifies that the output of the APSE program associated with the specified window is to be suspended.

2. WINDOW_NAME => <window name>

   Specifies the window whose APSE program's output is to be suspended.

4.2.30.4 Examples

4.2.30.4.1 Long Form

```
AIM> SUSPEND OBJECT_TYPE => PROGRAM_OUTPUT WINDOW_NAME => WIN_1
```

Suspend the output of the APSE program associated with WIN_1.

4.2.30.4.2 Mixed

```
AIM> SUSPEND PROGRAM_OUTPUT WINDOW_NAME => WIN_2
```

Suspend the output of the APSE program associated with WIN_2.

4.2.30.4.3 Short Form

```
AIM> SU E WIN_3
```

Suspend the output of the APSE program associated with WIN_3.

4.2.30.5 Errors

The semantic errors associated with this command include:

1. "Window <window name> does not exist"

   The specified window name, <window name>, does NOT exist (i.e.--no window of that name has been created)

4.2.31 TERMINATE EXECUTION

4.2.31.1 Functional Description

The TERMINATE EXECUTION command directs the AIM to TERMINATE execution of the program in the specified window. If there is not a program running in the specified window, no action is taken.

4.2.31.2 Syntax

```
T*ERMINATE  OBJECT_TYPE => E*XECUTION,
             WINDOW_NAME => <window name>
```

4.2.31.3 Command Parameters

1. OBJECT_TYPE => E*XECUTION

   Specifies that the execution of the APSE program associated with the specified window is to be terminated.

2. WINDOW_NAME => <window name>

   Specifies the window whose associated APSE program execution is to be terminated.

4.2.31.4 Examples

4.2.31.4.1 Long Form

```
AIM> TERMINATE OBJECT_TYPE => EXECUTION WINDOW_NAME => WIN_1
```

TERMINATE the execution of the APSE program associated with WIN_1.

4.2.31.4.2 Mixed

```
AIM> TERMINATE EXECUTION WINDOW_NAME => WIN_2
```

TERMINATE the execution of the APSE program associated with WIN_2.

4.2.31.4.3 Short Form

```
AIM> T E WIN_3
```

TERMINATE the execution of the APSE program associated with WIN_3.

4.2.31.5 Errors

The semantic errors associated with this command include:

1. "No subordinate process exists in window: <window_name>"

   There was no process to terminate in the specified window

4.3 KEYSTROKE COMMANDS

This section describes the Keystroke Commands available in the AIM. Because the keystrokes are terminal dependent the user is directed to find the format and examples of a particular keystroke in the appendices for the terminal type being used.

There are also no errors associated with the keystrokes.

4.3.1 ABORT_SCRIPT

4.3.1.1 Functional Description

The ABORT_SCRIPT command allows the user to abort a script. This command will abort all scripts that are currently running. If the scripts are nested, all scripts are aborted regardless of which script is executing when the abort keystroke is entered.

4.3.1.2 Format/Examples

See Appendix A for examples using your specific terminal type.

4.3.2 CLEAR_WINDOW

4.3.2.1 Functional Description

The CLEAR_WINDOW keystroke command allows the user to blank the current window of all information except the header.

4.3.2.2 Format/Examples

See Appendix A for examples using your specific terminal type.

4.3.3 NEXT_IMAGE

4.3.3.1 Functional Description

The NEXT_IMAGE keystroke command switches the cursor from the current image to the one that would follow if all the images were put in an alphabetized list. If none follows, the cursor moves to the image that would be first in the alphabetized list.

4.3.3.2 Format/Examples

See Appendix A for examples using your specific terminal type.

4.3.4 NEXT_PAGE

4.3.4.1 Functional Description

The NEXT_PAGE keystroke command displays the next output page. This command is used when the "More.." flag is set indicating that more text is available than is currently displayed.

4.3.4.2 Format/Examples

See Appendix A for examples using your specific terminal type.

4.3.5 NEXT_VIEWPORT

4.3.5.1 Functional Description

The NEXT_VIEWPORT keystroke command switches the cursor from the current viewport to the one that follows. If none follows, the cursor moves to the viewport at the top of the image.

4.3.5.2 Format/Examples

See Appendix A for examples using your specific terminal type.

4.3.6 PREVIOUS_IMAGE

4.3.6.1 Functional Description

The PREVIOUS_IMAGE keystroke command switches the cursor from the current image to the one that would precede it if all the images were put in an alphabetized list. If there is no previous image, the cursor moves to the image that would be last in the alphabetized list.

4.3.6.2 Format/Examples

See Appendix A for examples using your specific terminal type.

4.3.7 PREVIOUS_VIEWPORT

4.3.7.1 Functional Description

The PREVIOUS_VIEWPORT keystroke command switches the cursor from the viewport, where it currently exists, to the previous viewport on the image. If there is none, the cursor moves to the viewport that is last on the image.

4.3.7.2 Format/Examples

See Appendix A for examples using your specific terminal type.

4.3.8 REDISPLAY_SCREEN

4.3.8.1 Functional Description

The REDISPLAY_SCREEN keystroke command will redisplay the entire screen. This command is useful in cases where system messages are written over the information on the screen and the user wishes to have the original data redisplayed.

4.3.8.2 Format/Examples

See Appendix A for examples using your specific terminal type.

4.3.9 RETURN_TO_PREVIOUS_IMAGE

4.3.9.1 Functional Description

The RETURN_TO_PREVIOUS_IMAGE keystroke returns the cursor to the previous image. This keystroke differs from the PREVIOUS_IMAGE keystroke in that this return is not based on alphabetical ordering. Instead, with this keystroke, the cursor simply returns whatever image it was previously in. If this command is entered as the first entry into the AIM image, nothing will happen.

4.3.9.2 Format/Examples

See Appendix A for examples using your specific terminal type.

4.3.10 SWITCH_TO_AIM

4.3.10.1 Functional Description

The SWITCH_TO_AIM keystroke command transfers control to the AIM image. If the SUSPEND_ON_AIM flag is set, then all output to other windows is suspended until the user resumes output to those windows. The default is no suspension on SWITCH_TO_AIM, which means output to windows will continue asynchronously while the user interacts with the AIM image.

4.3.10.2 Format/Examples

See Appendix A for examples using your specific terminal type.

CHAPTER 5

AIM OPERATING INSTRUCTIONS

This chapter is a presentation of the conditions surrounding the operation of the AIM computer program within the Data General Ada Development Environment(ADE). The first section discusses the steps necessary to properly execute the AIM program; the second and third sections respectively discuss the AIM's expected input and output.

5.1 OPERATING PROCEDURES

This section discusses the setup, invocation, and termination of the AIM program.

5.1.1 AIM Setup

The following ADE files must be visible from the current working directory (directly or via the current searchlist) prior to the invocation of the AIM program:

1. AIM Executable Image(AIM.PR) -- the AIM's executable program image.

2. Terminal Identification File(TERM) -- file containing the name of the user's terminal to be used with the AIM program.

3. Terminal Capabilites File(TCF) -- the file containing a tablular description of the capability of specific ANSI terminals that will be used in conjuntion with the AIM program.

4. AIM Help File(AIM HELP FILE) -- the AIM's on-line Help file describing the AIM's textual and keystroke commands.

5. AIM Parse Table File(AIM PARSE TABLE FILE) -- the AIM parser input tables automatically generated by the NYU LALR Parser Generator when given a BNF description of the AIM grammar.

6. <u>AIM Initialization Script File(AIM INIT SCRIPT FILE)</u> -- an optional AIM command script file that is executed, if it exists, as part of the AIM initialization.

<u>Note</u>: the format and content of these files will be presented in the Input Section(5.2) of this chapter.

5.1.2 AIM Invocation

Once all of the AIM setup requirements have been met, the AIM can be invoked by the user via the ADE execute command:

```
-) XECUTE AIM
```

5.1.3 AIM Termination

The AIM ceases processing and returns to the ADE upon user request via the ABORT_AIM or EXIT command. The ABORT_AIM command effects an unconditional termination of the AIM program irregardless of any subordinate programs running underneath the AIM. The EXIT command will terminate the execution of the AIM program, if and only if, the AIM does not have any subordinate (grandson) processes; otherwise, a warning error will be generated and the AIM continues its execution.

5.2 AIM INPUTS

The AIM has seven areas where input is necessary:

1. Keyboard,

2. Terminal Identification File,

3. Terminal Capabilities File,

4. AIM Help File,

5. AIM Parse Table File,

6. AIM Initialization Script File, and

7. User defined AIM Script Files.

A discussion of each follows.

5.2.1 Keyboard

5.2.1.1 Purpose And Use

Keyboard input allows the AIM user to enter AIM commands and ADE program data.

5.2.1.2 Input Media

The source of the AIM's keyboard input is the user's terminal keyboard.

5.2.1.3 Format

The AIM's keyboard input can take the form of a stream of single characters, or a full line of information.

5.2.1.4 Content

The AIM's keyboard input contains AIM commands and input data for APSE programs within the ADE.

5.2.2 Terminal Identification File

5.2.2.1 Purpose And Use

The user's terminal name is identified by using standard Ada TEXT_IO. The terminal name is read in from the TERM file.

5.2.2.2 Input Media

The user's terminal name is stored in an ADE database file named TERM.

5.2.2.3 Format

The TERM file is a single line ASCII text file specifying the (case sensitive) name of the user's terminal beginning in column 1.

5.2.2.4 Content

A bug in the run-time support of the ADE caused the END_OF_FILE exception to be raised prematurely when reading the TERM file. Due to this, a dummy line has to be placed after the line containing the terminal name. An example file would look like:

```
tv970
dummy line
```

where "tv970" is the terminal name to be used with the AIM program.

5.2.3 Terminal Capabilities File

5.2.3.1 Purpose And Use

The Terminal Capabilities File (TCF) is a data base of terminal capability descriptions. Terminals are described in the TCF by giving a set of capabilities which they have, and by describing how operations are performed.

When invoked, the AIM opens the TCF, searches for the terminal name specified in the TERM file, and initializes the Virtual Terminal data structures with the corresponding terminal's capabilities.

5.2.3.2 Input Media

The AIM Terminal Capabilities File is stored in an ADE database file named TCF.

5.2.3.3 Format

The Terminal Capabilities File is a variation of the TERMCAP developed in the Berkeley extensions to the UNIX operating system. It contains a series of entries that describe the capabilities of various terminals. Each entry is composed of fields separated by colons (':'); multiple lines can be included in an entry by placing a '\' at the end of each line except for the last one of the entry.

The first entry for each terminal gives the names which are known for the terminal, separated by "|" characters. The first name is always 2 characters long and is used by older UNIX systems which store the terminal type in a 16 bit word in a system-wide data base. The second name given is the most common abbreviation for the terminal, and the last name given should be a name fully identifying the terminal. This second name is the one that is matched against the contents of the TERM file (note that this string search is case sensitive). The second name should contain no blanks; the last name may well contain blanks for readability.

The remainder of the fields are specified via 2 letter codes coupled with optional value fields. A description of all the two letter mnemonics follows:

(P) indicates padding may be specified.
Padding is an amount of time to be waited after the command is executed.

Name	Type	Pad?	Description
al	str	(P)	Add new blank line
be	str	(P)	Bell
cd	str	(P)	Clear to end of display
co	num		Number of columns in a line
ce	str	(P)	Clear to end of line
cm	str	(P)	Cursor motion
dc	str	(P)	Delete character
dl	str	(P)	Delete line
ei	str		End insert mode
g0	str		sent by terminal function key 20

g1	str		sent by terminal function key 21
g2	str		Sent by terminal function key 22
g3	str		Sent by terminal function key 23
g4	str		Sent by terminal function key 24
g5	str		Sent by terminal function key 25
g6	str		Sent by terminal function key 26
g7	str		Sent by terminal function key 27
g8	str		Sent by terminal function key 28
g9	str		Sent by terminal function key 29
h0	str		Label on function key 20
h1	str		Label on function key 21
h2	str		Label on function key 22
h3	str		Label on function key 23
h4	str		Label on function key 24
h5	str		Label on function key 25
h6	str		Label on function key 26
h7	str		Label on function key 27
h8	str		Label on function key 28
h9	str		Label on function key 29
im	str		Enter insert character mode
is	str		Terminal initialization string
k1	str		Sent by terminal function key 1
k2	str		Sent by terminal function key 2
k3	str		Sent by terminal function key 3
k4	str		Sent by terminal function key 4
k5	str		Sent by terminal function key 5
k6	str		Sent by terminal function key 6
k7	str		Sent by terminal function key 7
k8	str		Sent by terminal function key 8
k9	str		Sent by terminal function key 9
l1	str		Label on function key 1
l2	str		Label on function key 2
l3	str		Label on function key 3
l4	str		Label on function key 4
l5	str		Label on function key 5
l6	str		Label on function key 6
l7	str		Label on function key 7
l8	str		Label on function key 8
l9	str		Label on function key 9
kd	str		Sent by terminal down arrow key
kl	str		Sent by terminal left arrow key
kr	str		Sent by terminal right arrow key
ku	str		Sent by terminal up arrow key
li	num		Number of lines on screen or page
nl	str	(P)	Newline character (default \n)
se	str		End stand out mode
sf	str	(P)	Scroll forwards
so	str		Begin stand out mode
sr	str	(P)	Scroll reverse (backwards)
su	bool		Scrolls up at bottom of screen

t0	str	Sent by terminal function key 30
t1	str	Sent by terminal function key 31
t2	str	Sent by terminal function key 32
v0	str	Label on function key 30
v1	str	Label on function key 31
v2	str	Label on function key 32
wr	bool	Wraps at end of line
x0	str	Sent by terminal function key 10
x1	str	Sent by terminal function key 11
x2	str	Sent by terminal function key 12
x3	str	Sent by terminal function key 13
x4	str	Sent by terminal function key 14
x5	str	Sent by terminal function key 15
x6	str	Sent by terminal function key 16
x7	str	Sent by terminal function key 17
x8	str	Sent by terminal function key 18
x9	str	Sent by terminal function key 19
y0	str	Label on function key 10
y1	str	Label on function key 11
y2	str	Label on function key 12
y3	str	Label on function key 13
y4	str	Label on function key 14
y5	str	Label on function key 15
y6	str	Label on function key 16
y7	str	Label on function key 17
y8	str	Label on function key 18
y9	str	Label on function key 19

It is obvious from this list that there are three basic types of terminal capability fields: Boolean capabilities which indicate that the terminal has some particular feature, numeric capabilities giving the size of the terminal or the size of particular delays, and string capabilities, which give a sequence which can be used to perform particular terminal operations. Note that these 2 letter codes may appear in any order within a temrinal's TCF entry.

Numeric capabilities are followed by the character "#" and then the value. Thus "co" which indicates the number of columns the terminal has gives the value "80" for the Televideo-970.

String valued capabilities, such as "ce" (clear to end of line sequence) are given by the two character code, an "=", and then a string ending at the next following ":". A delay in milliseconds may appear after the "=" in such a capability. The delay must be an integer, e.g. "20".

A number of escape sequences are provided in the string valued capabilities for easy encoding of characters there. A "\E" maps to an ESCAPE character, "^x" maps to a control-x for any appropriate x, and the sequences "\n" "\r" "\t" "\b" "\f" give a newline, return, tab, backspace and form-feed respectively. Finally, characters may be given as three octal digits after a "\", and the characters "^" and "\" may be given as "\^" and "\\". If it is necessary to place a ":" in a capability it must be escaped in octal as "\072".

5.2.3.4 Content

The Terminal Capabilities File is an ADE database file which contains mappings from logical computer terminal characteristics to specific computer terminal characteristics. Typically, the TCF contains information that supports many different terminal types and makes. The following Terminal Capabilities File entry describes the capabilities of a Televideo-970:

```
tl|tv970|tv-970|televideo 970:\
        :al=1\E[1L:cd=\E[J:ce=\E[K:cm=\E[%i%2;%2H:co#80:\
        :dc=\E[1P:dl=1*\E[1M:ei=\E[41:im=\E[4h:li#24:\
        :se=\E[0m:so=\E[7m:\
        :kb=^h:ku=\E[A:kd=\E[B:kl=\E[D:kr=\E[C:\
        :kl=\E?a\014:l1=F1:k2=\E?b\014:l2=F2:\
        :k3=\E?c\014:l3=F3:k4=\E?d\014:l4=F4:k5=\E?e\014:l5=F5:\
        :k6=\E?f\014:l6=F6:k7=\E?g\014:l7=F7:k8=\E?h\014:l7=F7:\
        :k9=\E?i\014:l9=F9:x0=\E?j\014:y0=F10:x1=\E?k\014:y1=F11:\
        :x2=\E?l\014:y2=F12:x3=\E?m\014:y3=F13:x4=\E?n\014:y4=F14:\
        :x5=\E?o\014:y5=F15:x6=\E?p\014:y6=F16:\
        :sr=\EM:is=\E<\E[>1;2;3;4;5;6;7;8;91\E[0m\E[11m\E[?7h:\
        :be=\E[?5h\E[?51:
```

5.2.4 AIM Help File

5.2.4.1 Purpose And Use

During the execution of the AIM, the user may request on-line help for AIM related topics. The information supporting on-line Help is contained in the AIM's help file.

5.2.4.2 Input Media

The AIM Help File is stored in an ADE database file named AIM_HELP_FILE.

5.2.4.3 Format

The text file used by the HELP_UTILITY Package is required to have a particular format. If the file is not in this format, an exception will be raised. The following explains the required format.

COMMENTS: Comments may be embeded in the text file. All comments are ignored when the file is read into memory. A text line is considered a comment if the first and second characters of the line are minus signs (--).

TOPICS: The first non-comment text line MUST begin with the digit 1 in column one. This number is the topic level. In other words, text (as defined below) cannot be found in the Help file before a topic is found to which the text can be associated. Topics are those subjects for which information is being provided. A topic name may contain any printable character except blanks. Embedded blanks are NOT allowed in a topic name. This will not be flagged as an error but the full name will not be recognized within the AIM. All letters in the name must be capitals. It is not required to have a space separating the topic level from the topic name. Any line beginning with a digit will be considered a topic line.

SUBTOPICS: A topic may have subtopics. Subtopics are denoted by having a level exactly one greater than the associated topic level. Subtopics follow the same rules as topics in all other aspects. There is no constraint (other than a lack of memory) on the number of subtopic levels (i.e.--subtopics may have subtopics).

TEXT: All text lines not beginning with two consecutive minus signs or a digit will be considered text.

The text file is saved exactly as the user sees it (including blank lines) with the follow exceptions:

- o topic and subtopic names have leading blanks stripped off,
- o if the topic or subtopic name is longer than one half the screen size, it may be truncated when a menu of information is output,
- o if the text line is longer than the screen size, the text line is truncated before output, and
- o the text file lines are assumed to be eighty characters maximum.

5.2.4.4 Content

The AIM Help file contains help messages for all AIM related topics. A brief sample of the Help file's overall content follows:

```
1 ASSOCIATE
  The ASSOCIATE command maps a specified portion of a window onto an
  image. Assuming that n = <length> and p = <position> the ASSOCIATE
  command maps the last n lines of the specified window onto the given
  image starting at the pth line of the image.  For example,
ASSOC WIN_1 IM_1 13 12
```

will map the last 12 lines of WIN_1 onto lines 13..24 of IM_1. This forms an abstract association between WIN_1 and IM_1 by creating a viewport (lines 13 thru 24) onto which the last 12 lines of WIN_1 are mapped. This viewport partitions the image vertically, and its width is equal to the width of the image.

Note: A window may be associated with more than one image at the same time; however, a specific window may only be associated with a specific image once.

2 SYNTAX

```
AS*SOCIATE WINDOW_NAME => <window_name>
           IMAGE_NAME  => <image_name>
           TOP         => <position>
           LENGTH      => <length>
```

2 COMMAND_PARAMETERS

WINDOW_NAME => <window_name>

Specifies the window whose last <length> lines are to be mapped onto the specified image.

IMAGE_NAME => <image_name>

Specifies the image onto which a portion of the given window will be mapped.

TOP => <position>

Specifies the starting position for the association relative to the top of the specified image.

LENGTH => <length>

Specifies the length of the viewport used for the requested association. In addition to the viewport header, at least one line of a window must be displayed in a viewport; therefore, the minimum length allowable is 2.

2 EXAMPLES

```
Long Form:       AIM> ASSOCIATE WINDOW_NAME => W_1
                                 IMAGE_NAME => I_1
                                        TOP => 1
                                     LENGTH => 24
```

Associate the entire contents of window W_1 with the image I_1.

```
Mixed Form:      AIM> ASSOC WIN_2 IMAGE_NAME => IM_2 1 LENGTH => 8
```

Associate the last 8 lines of window WIN_2 with the first 8 lines of image IM_1.

Short Form: `AIM> AS WIN_3 IM_3 9 8`

Associate the last 8 lines of window WIN_3 with lines 9 thru 16 of image IM_1.

2 ERRORS

The semantic errors associated with this command include:

"Window <window_name> does not exist"

The specified window name, <window_name> does NOT exist (i.e.-- no window of that name has been created).

"Image <image_name> does not exist"

The specified image name, <image_name> does NOT exist (i.e.-- no image of that name has been created).

"Invalid length: <length>"

The specified viewport length was either out of range, non-numeric, or too long to ensure non-intersecting viewports of the given image

"Invalid top line: <top_line>"

The specified top line for starting the viewport was either out of range, or non-numeric.

"Association between specified window and image already exists"

The specified window is currently associated with <image_name> and the AIM prohibits multiple associations between the same window and image.

"Cannot map another window onto the AIM image"

The AIM image and window cannot be altered by the user.

5.2.5 AIM Parse Table File

5.2.5.1 Purpose And Use

The AIM Parse Table File is used by the AIM's command line interpreter (CLI) to parse incoming AIM textual commands. These LALR parse tables are read in by the AIM's CLI during initialization and used by the AIM parser throughout the AIM's execution.

5.2.5.2 Input Media

The AIM Parse Table File is stored in an ADE database file named AIM_PARSE_TABLE_FILE.

5.2.5.3 Format

A description of the LALR parsing tables extracted from the NYU LALR Parser Generator User's Manual [NYU81] follows.
The LALR Parse Table file contains the following 8 objects:

(1) Map of symbols used in the grammar. (NOSYM)
(2) Tuple of left hand side symbol. (LHS)
(3) Tuple of size of right hand sides. (RHS)
(4) Action table 1. (see below)
(5) Action table 2. (see below)
(6) Tuple of default reductions. (DEFAULT)
(7) Tuple of symbols on which shifts were made to branch into a new state. (IN_SYM)
(8) Follow map. (FOLLOW)

The first line of the output file contains the size of the 8 objects together with some constants needed to parse a source input. The format is 12I6 and the fields are:

(1)	Number of symbols.	
(2)	Number of left hand sides.	
(3)	Number of right hand side sizes.	
(4)	Number of entries in Action Table 1.	
(5)	Number of entries in Action Table 2.	
(6)	Number of entries in Default tuple.	
(7)	Number of entries in IN_SYM.	
(8)	Number of entries in follow map.	
(9)	Number of States + 1.	(NUM_STATES)
(10)	Number of symbols + 1.	(NUM_INPUTS)
(11)	Number of Actions.	(NUM_ACTIONS)
(12)	Table size.	(TABLE_SIZE)

The encoding of the objects is as follows:

(1) NOSYM map: (I4, I2, 64A1) format.
The three fields are:
a) Numeric value assigned to the symbol
b) size of the string representing the symbol
c) the string representing the symbol

(2,6,7) LHS, DEFAULT, and IN_SYM tuples: 18I4 format

(3) RHS tuple : 36I2 format.

(4,5) ACTION_TABLE1 and ACTION_TABLE2 tuples: 9I8 format

(8) FOLLOW map: 18I4 format. Each entry in the map begins on a new line. The first value is the domain value, the second is the size of the range value(which is a set)

and the remaining values are the elements in this set. If there are more than 16 elements in the set, they are printed on successive lines.

Each object begins on a new line. Thus at the end of an object, the last line may not be filled up.

5.2.5.4 Content

Given the following simple grammar specification:

```
1  E ::= E + T
2  E ::= T
3  T ::= T * F
4  T ::= F
5  F ::= ( E )
6  F ::= ID
```

the LALR Parser Generator produces the following as its parse table file:

```
   11     6     6    43    43    12    12     4    13    12    21    37
  1 0
  2 1+
  3 1*
  4 1(
  5 1)
  6 2ID
  7 4$EOF
  8 4$ACC
  9 1E
 10 1T
 11 1F
  9   9  10  10  11  11
 3 1 3 1 3 1
      0      55       0      61       0       4       0       0      58
      6       2       0       8       0       0      10       3       0
      4       0       0       5       6       2       9       8       0
      0       4       0      13       0      11       2       3       9
      0      12       7       0       4       3       0
      0       0       0       0       0      42       0       0       0
     46      47       0      86       0       0      89      16       0
     18       0       0      21      22      23     135      62     100
      0     102       0      67       0     106     107     112      75
      0     119      45       0     114      40       0
  0  17   0  19   0  15   0   0   0  18  14  16
  1  11   4   6   9  10   9   2   3   5  10  11
  8   1   7
  9   3   2   5   7
```

10	4	2	3	5	7
11	4	2	3	5	7

5.2.6 AIM Initialization Script File

5.2.6.1 Purpose And Use

Upon invocation, the AIM attempts to open and read a pre-defined initialization script file. If the file exists, the AIM command interpreter sets the AIM window to be the currently active window, and then reads and executes the commands present in the file; otherwise, the AIM CLI sets the MAIN window to be the currently active window.

5.2.6.2 Input Media

The AIM initialization script file is stored in an ADE database file named AIM_INIT_SCRIPT_FILE.

5.2.6.3 Format

The AIM initialization script file is an ASCII text file of AIM textual commands.

5.2.6.4 Content

The AIM initialization script file contains AIM textual command lines. For example:

```
    DEFINE TERMINAL "TV970"
    RESET AIM_SUSPENDS
    RESET FULL AIM
    RESET FULL MAIN
```

This initialization script file defines the terminal to be a TeleVideo 970 and also resets various AIM flags.

5.2.7 User Defined Script Files

5.2.7.1 Purpose And Use

During the execution of the AIM program a user may use the AIM "SCRIPT" command to read and interpret a group of AIM commands from a text file. Using AIM script files provides a quick mechanism for re-creating frequently used window/image scenarios.

5.2.7.2 Input Media

An AIM script file is stored in ADE database file whose name is defined by the user and is not to exceed 20 charcaters in length.

5.2.7.3 Format

An AIM script file is an ASCII text file of AIM textual commands.

5.2.7.4 Content

An AIM script file contains AIM textual command lines. For example:

```
CREATE IMAGE     IMAGE_1
CREATE IMAGE     IMAGE_2
CREATE WINDOW    WINDOW_A
CREATE WINDOW    WINDOW_B
CREATE WINDOW    WINDOW_C
CREATE WINDOW    WINDOW_D
ASSOC  WINDOW_A  IMAGE_1  1   8
ASSOC  WINDOW_B  IMAGE_1  9   8
ASSOC  WINDOW_C  IMAGE_1  17  8
ASSOC  WINDOW_C  IMAGE_2  1   12
ASSOC  WINDOW_D  IMAGE_2  13  12
```

This AIM script file creates two images and four windows, and then maps these windows onto the given images.

5.3 AIM OUTPUTS

The AIM has three areas where output is generated:

1. Terminal Output,

2. Input Pad Files, and

3. Output Pad Files.

A discussion of each follows.

5.3.1 Terminal Output

5.3.1.1 Purpose And Use

The AIM generates terminal output in response to the interpretation of AIM commands, and as a direct result of executing the users ADE programs.

5.3.1.2 Output Media

The AIM generates interactive output to the user's terminal.

5.3.1.3 Format

The AIM output to the user's terminal is a stream of characters.

5.3.1.4 Content

The terminal output generated by the AIM includes: output from the AIM's command interpreter(Command Prompts, On-line Help information, AIM Environmental Information, Error Messages), and output from ADE programs.

5.3.2 Input Pad Files

5.3.2.1 Purpose And Use

An AIM user can at any time request the activation of a window's input pad. Each activation of an input pad creates a unique ADE database file. When a window's input pad is active, all input destined for the particular window is also written to the corresponding input pad file.

5.3.2.2 Output Media

An AIM input pad is stored in an ADE database file.

5.3.2.3 Format

An AIM input pad file is an ASCII text file.

5.3.2.4 Content

An AIM input pad for a window named X contains all the input destined for X during the time period that the input pad was active.

5.3.3 Output Pad Files

5.3.3.1 Purpose And Use

An AIM user can at any time request the activation of a window's output pad. Each activation of an output pad creates a unique ADE database file. When a window's output pad is active, all output destined for the particular window is also written to the corresponding output pad file.

5.3.3.2 Output Media

An AIM output pad is stored in an ADE database file.

5.3.3.3 Format

An AIM output pad file is an ASCII text file.

5.3.3.4 Content

An AIM output pad for a window named X contains all the output destined for X during the time period that the output pad was active.

5.3.4 AIM Input-Output Diagram

An overall Input-Output Diagram of the AIM is shown in the following figure.

```
                                    +----------+
         Keyboard Input --->        |          |
                                    |          |
   Terminal Identification ->       |          | ---> Terminal Output
        File (TERM)                 |          |
                                    |          |
   Terminal Capabilities --->       |          |
        File (TCF)                  |          |
                                    |   AIM    |
         Command Script --->        |          |  ---> Input Pad(s)
             File(s)                |          |
                                    |          |
          HELP file     --->        |          |
                                    |          |
          AIM Parse     --->        |          |  ---> Output Pad(s)
          Table File                |          |
                                    |          |
                                    +----------+

           INPUT                                      OUTPUT
```

Figure 5.1--AIM Input-Output Diagram

APPENDIX A

PAGE TERMINAL TUTORIAL

The following are sample AIM sessions. Each is independently complete in that it assumes nothing about the prior AIM context. The AIM is invoked at the beginning of each session, and subsequently exited at the end of each session.

Session Assumptions:

1. Data General Corporation's Ada Development Environment

2. a Page terminal

3. terminal display with 24 lines and 80 character columns

For the reader's convenience, the following conventions will be used in the presentation of this session:

1. ALL user responses will appear on the last nonblank line of the screen

2. the affect of executing a user's command will be presented in the next screen relative to when the command was issued

3. the comments below each screen will explain:

   a. the affect of executing the previous AIM command

   b. the contents of the current screen

   c. the new AIM command issued at the bottom of the present screen

The default key sequences that exist when the AIM is invoked from a ASCII Page Mode Terminal are as follows:

DEFAULT KEYSTROKES

ABORT_SCRIPT	F1
CLEAR_WINDOW	F2
NEXT_IMAGE	F3
NEXT_PAGE	F4
NEXT_VIEWPORT	F5
PREVIOUS_IMAGE	F6
PREVIOUS_VIEWPORT	F7
REDISPLAY_SCREEN	F8
RETURN_TO_PREVIOUS_IMAGE	F9
SWITCH_TO_AIM	F10

where "F#" represents a function key.

A.1 SESSION 1 - BASIC AIM COMMANDS

The following AIM command session demonstrates the use of some basic AIM commands, including:

* CREATE
* ASSOCIATE
* GOTO
* DELETE
* DISASSOCIATE
* EXIT

```
                              INVOKE AIM FROM APSE
________________________________________________________________________________
) AIM




























_____________________________BOTTOM-OF-SCREEN___________________________________
```

Comments:

Command: AIM - invoke the AIM from the APSE.

SWITCH TO AIM IMAGE

```
 MAIN            MAIN         FRCH:016        Running                      AF
AOS/VS CLI REV 05.01.00.00           9-APR-85      13:26:43 {F10}








__________________________BOTTOM-OF-SCREEN_______________________________
```

Comments: The AIM initializes the current image to be MAIN. To enter AIM textual commands the user must communicate through the AIM window to the AIM command interpreter.

Command: {F10} - Set the current image to be the AIM image.

CREATE A NEW IMAGE

```
 AIM              AIM       AIM CLI        Running                        AF
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1
```

```
__________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The AIM Command Interpreter is awaiting input from the terminal.

Command: CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1 - Create an image named IMAGE_1.

CREATE A NEW WINDOW

```
 AIM               AIM       AIM CLI       Running                                AF
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WINDOW_A
```

```
___________________________BOTTOM-OF-SCREEN________________________________
```

Comments: The AIM Command Interpreter has created a new image named IMAGE_1.

Command: CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WINDOW_A - Create a window named WINDOW_A.

CREATE A NEW WINDOW

```
 AIM               AIM      AIM CLI      Running                               AF
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WINDOW_A
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_B
```

```
_______________________________BOTTOM-OF-SCREEN_______________________________
```

Comments: The AIM Command Interpreter has created a new window named WINDOW_A.

Command: CREATE OBJECT_TYPE => WINDOW WINDOW_B - Create a window named WINDOW_B.

CREATE A NEW WINDOW

```
 AIM             AIM       AIM CLI       Running                         AF
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WINDOW_A
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_B
AIM> CR WINDOW WINDOW_C










__________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The AIM Command Interpreter has created a new window named WINDOW_B.

Command: CR WINDOW WINDOW_C - Create a window named WINDOW_C.

ASSOCIATE WINDOW_A WITH IMAGE_1

```
 AIM                 AIM         AIM CLI        Running                               AF
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WINDOW_A
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_B
AIM> CR WINDOW WINDOW_C
AIM> ASSOC WINDOW_A IMAGE_1 1 8
```

```
____________________________________BOTTOM-OF-SCREEN_______________________________________
```

Comments: The AIM Command Interpreter has created a new window named WINDOW_C.

Command: ASSOC WINDOW_A IMAGE_1 1 8 - Map the last 8 lines of WINDOW_A onto the first 8 lines of IMAGE_1.

ASSOCIATE WINDOW_B WITH IMAGE_1

```
 AIM              AIM       AIM CLI       Running                        AF
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WINDOW_A
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_B
AIM> CR WINDOW WINDOW_C
AIM> ASSOC WINDOW_A IMAGE_1 1 8
AIM> ASSOC WINDOW_B IMAGE_1 9 8
```

____________________BOTTOM-OF-SCREEN____________________

Comments: The last 8 lines of WINDOW_A are associated with the first 8 lines of IMAGE_1.

Command: ASSOC WINDOW_B IMAGE_1 9 8 - Map the last 8 lines of WINDOW_B onto the middle(9..16) 8 lines of IMAGE_1.

ASSOCIATE WINDOW_C WITH IMAGE_1

```
 AIM              AIM      AIM CLI        Running                              AF
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WINDOW_A
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_B
AIM> CR WINDOW WINDOW_C
AIM> ASSOC WINDOW_A IMAGE_1 1 8
AIM> ASSOC WINDOW_B IMAGE_1 9 8
AIM> ASSOC WINDOW_C IMAGE_1 17 8
```

```
___________________________BOTTOM-OF-SCREEN___________________________
```

Comments: WINDOW_A and WINDOW_B are currently mapped onto IMAGE_1.

Command: ASSOC WINDOW_C IMAGE_1 17 8 - Map the last 8 lines of WINDOW_C onto the last(17..24) 8 lines of IMAGE_1.

GOTO IMAGE IMAGE_1

```
 AIM             AIM       AIM CLI       Running                                AF
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WINDOW_A
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_B
AIM> CR WINDOW WINDOW_C
AIM> ASSOC WINDOW_A IMAGE_1 1 8
AIM> ASSOC WINDOW_B IMAGE_1 9 8
AIM> ASSOC WINDOW_C IMAGE_1 17 8
AIM> GOTO IMAGE IMAGE_1









___________________________BOTTOM-OF-SCREEN________________________________
```

Comments: WINDOW_A, WINDOW_B, and WINDOW_C are currently mapped onto IMAGE_1.

Command: GOTO IMAGE IMAGE_1 - Position the cursor in the top viewport of IMAGE_1.

```
                    CURRENT IMAGE ON SCREEN IS IMAGE_1
______________________________________________________________________________
 IMAGE_1         WINDOW_A        FRCH:017        Running                   AF

AOS/VS CLI   REV 05.01.00.00         11-APR-85    8:55:29 {F10}



 IMAGE_1         WINDOW_B        FRCH:018        Running                   AF

AOS/VS CLI   REV 05.01.00.00         11-APR-85    8:55:29



 IMAGE_1         WINDOW_C        FRCH:019         Running                   AF

AOS/VS CLI   REV 05.01.00.00         11-APR-85    8:55:29


______________________________BOTTOM-OF-SCREEN________________________________
```

Comments: The cursor was positioned in the first viewport of IMAGE_1. Each window is executing its own APSE process, namely the APSE command interpreter; thus, the APSE command interpreter prompt, ")", appears in WINDOW_A, WINDOW_B, and WINDOW_C. Note, each viewport header displays the default flag settings for the windows ("AF").

Command: {F10} - Switch control back to the AIM command interpreter in the AIM window.

GOTO WINDOW WINDOW_C

```
 AIM             AIM      AIM CLI       Running                        AF
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WINDOW_A
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_B
AIM> CR WINDOW WINDOW_C
AIM> ASSOC WINDOW_A IMAGE_1 1 8
AIM> ASSOC WINDOW_B IMAGE_1 9 8
AIM> ASSOC WINDOW_C IMAGE_1 17 8
AIM> GOTO IMAGE IMAGE_1
AIM> GOTO WINDOW WINDOW_C
```

BOTTOM-OF-SCREEN

Comments: Control has returned to the AIM command interpreter.

Command: GOTO WINDOW WINDOW_C - Position the cursor in the largest viewport associated with WINDOW_C.

CURRENT IMAGE ON SCREEN IS IMAGE_1

```
 IMAGE_1          WINDOW_A         FRCH:017         Suspended                 AFS

AOS/VS CLI     REV 05.01.00.00          11-APR-85     8:55:29



 IMAGE_1          WINDOW_B         FRCH:018         Suspended                 AFS

AOS/VS CLI     REV 05.01.00.00          11-APR-85     8:55:29



 IMAGE_1          WINDOW_C         FRCH:019         Suspended                 AFS

AOS/VS CLI     REV 05.01.00.00          11-APR-85     8:55:29 {F10}



_______________________________BOTTOM-OF-SCREEN_______________________________
```

Comments: The cursor was positioned in the third viewport of IMAGE_1 since it was the largest one associated with WINDOW_C. Note that the suspend flag is set, this was caused by the Switch-to-Aim keystroke.

Command: {F10} Switch control back to the AIM command interpreter in the AIM window.

DISASSOCIATE WINDOW_A AND IMAGE_1

```
 AIM                  AIM       AIM CLI        Running                               AF
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WINDOW_A
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_B
AIM> CR WINDOW WINDOW_C
AIM> ASSOC WINDOW_A IMAGE_1 1 8
AIM> ASSOC WINDOW_B IMAGE_1 9 8
AIM> ASSOC WINDOW_C IMAGE_1 17 8
AIM> GOTO IMAGE IMAGE_1
AIM> GOTO WINDOW WINDOW_C
AIM> DIS WINDOW_A IMAGE_1









__________________________BOTTOM-OF-SCREEN__________________________
```

Comments: Control has returned to the AIM command interpreter.

Command: DIS WINDOW_A IMAGE_1 - "DIS" is a valid abbreviation for the DISASSOCIATE command; this command instructs the AIM command interpreter to break the logical association between WINDOW_A and IMAGE_1.

DISASSOCIATE WINDOW_B AND IMAGE_1

```
 AIM               AIM         AIM CLI        Running                           AF
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WINDOW_A
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_B
AIM> CR WINDOW WINDOW_C
AIM> ASSOC WINDOW_A IMAGE_1 1 8
AIM> ASSOC WINDOW_B IMAGE_1 9 8
AIM> ASSOC WINDOW_C IMAGE_1 17 8
AIM> GOTO IMAGE IMAGE_1
AIM> GOTO WINDOW WINDOW_C
AIM> DIS WINDOW_A IMAGE_1
AIM> DIS WINDOW_B IMAGE_1









________________________BOTTOM-OF-SCREEN________________________
```

Comments: The logical association between WINDOW_A and IMAGE_1 has been dissolved.

Command: DIS WINDOW_B IMAGE_1 - Break the logical association between WINDOW_B and IMAGE_1.

DISASSOCIATE WINDOW_C AND IMAGE_1

```
 AIM              AIM       AIM CLI       Running                        AF
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WINDOW_A
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_B
AIM> CR WINDOW WINDOW_C
AIM> ASSOC WINDOW_A IMAGE_1 1 8
AIM> ASSOC WINDOW_B IMAGE_1 9 8
AIM> ASSOC WINDOW_C IMAGE_1 17 8
AIM> GOTO IMAGE IMAGE_1
AIM> GOTO WINDOW WINDOW_C
AIM> DIS WINDOW_A IMAGE_1
AIM> DIS WINDOW_B IMAGE_1
AIM> DIS WINDOW_C IMAGE_1









__________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The logical association between WINDOW_B and IMAGE_1 has been dissolved.

Command: DIS WINDOW_C IMAGE_1 - Break the logical association between WINDOW_C and IMAGE_1.

DELETE WINDOW WINDOW_A

```
 AIM              AIM      AIM CLI       Running                         AF
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WINDOW_A
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_B
AIM> CR WINDOW WINDOW_C
AIM> ASSOC WINDOW_A IMAGE_1 1 8
AIM> ASSOC WINDOW_B IMAGE_1 9 8
AIM> ASSOC WINDOW_C IMAGE_1 17 8
AIM> GOTO IMAGE IMAGE_1
AIM> GOTO WINDOW WINDOW_C
AIM> DIS WINDOW_A IMAGE_1
AIM> DIS WINDOW_B IMAGE_1
AIM> DIS WINDOW_C IMAGE_1
AIM> DEL WINDOW WINDOW_A







________________________BOTTOM-OF-SCREEN________________________
```

Comments: The logical association between WINDOW_C and IMAGE_1 has been dissolved.

Command: DEL WINDOW WINDOW_A - "DEL" is a valid abbreviation for the AIM delete command. This command instructs the AIM command interpreter to remove WINDOW_A from the internal list of windows in the AIM environment.

DELETE WINDOW WINDOW_B

```
 AIM              AIM       AIM CLI        Running                                  AF
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WINDOW_A
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_B
AIM> CR WINDOW WINDOW_C
AIM> ASSOC WINDOW_A IMAGE_1 1 8
AIM> ASSOC WINDOW_B IMAGE_1 9 8
AIM> ASSOC WINDOW_C IMAGE_1 17 8
AIM> GOTO IMAGE IMAGE_1
AIM> GOTO WINDOW WINDOW_C
AIM> DIS WINDOW_A IMAGE_1
AIM> DIS WINDOW_B IMAGE_1
AIM> DIS WINDOW_C IMAGE_1
AIM> DEL WINDOW WINDOW_A
AIM> DEL WINDOW WINDOW_B





__________________________BOTTOM-OF-SCREEN_____________________________
```

Comments: WINDOW_A has been deleted from the AIM's internal list of windows.

Command: DEL WINDOW WINDOW_B - Remove WINDOW_B from the internal list of windows in the AIM environment.

DELETE WINDOW WINDOW_C

```
 AIM              AIM       AIM CLI        Running                          AF
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WINDOW_A
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_B
AIM> CR WINDOW WINDOW_C
AIM> ASSOC WINDOW_A IMAGE_1 1 8
AIM> ASSOC WINDOW_B IMAGE_1 9 8
AIM> ASSOC WINDOW_C IMAGE_1 17 8
AIM> GOTO IMAGE IMAGE_1
AIM> GOTO WINDOW WINDOW_C
AIM> DIS WINDOW_A IMAGE_1
AIM> DIS WINDOW_B IMAGE_1
AIM> DIS WINDOW_C IMAGE_1
AIM> DEL WINDOW WINDOW_A
AIM> DEL WINDOW WINDOW_B
AIM> DEL WINDOW WINDOW_C




__________________________BOTTOM-OF-SCREEN______________________________
```

Comments: WINDOW_B has been deleted from the AIM's internal list of windows.

Command: DEL WINDOW WINDOW_C - Remove WINDOW_C from the internal list of windows in the AIM environment.

DELETE IMAGE IMAGE_1

```
 AIM               AIM      AIM CLI       Running                               AF
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WINDOW_A
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_B
AIM> CR WINDOW WINDOW_C
AIM> ASSOC WINDOW_A IMAGE_1 1 8
AIM> ASSOC WINDOW_B IMAGE_1 9 8
AIM> ASSOC WINDOW_C IMAGE_1 17 8
AIM> GOTO IMAGE IMAGE_1
AIM> GOTO WINDOW WINDOW_C
AIM> DIS WINDOW_A IMAGE_1
AIM> DIS WINDOW_B IMAGE_1
AIM> DIS WINDOW_C IMAGE_1
AIM> DEL WINDOW WINDOW_A
AIM> DEL WINDOW WINDOW_B
AIM> DEL WINDOW WINDOW_C
AIM> DEL IMAGE IMAGE_1




__________________________BOTTOM-OF-SCREEN______________________________
```

Comments: WINDOW_C has been deleted from the AIM's internal list of windows.

Command: DEL IMAGE IMAGE_1 - Remove IMAGE_1 from the internal list of images in the AIM environment.

EXIT AIM

```
 AIM              AIM      AIM CLI       Running                          AF
AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IMAGE_1
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WINDOW_A
AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_B
AIM> CR WINDOW WINDOW_C
AIM> ASSOC WINDOW_A IMAGE_1 1 8
AIM> ASSOC WINDOW_B IMAGE_1 9 8
AIM> ASSOC WINDOW_C IMAGE_1 17 8
AIM> GOTO IMAGE IMAGE_1
AIM> GOTO WINDOW WINDOW_C
AIM> DIS WINDOW_A IMAGE_1
AIM> DIS WINDOW_B IMAGE_1
AIM> DIS WINDOW_C IMAGE_1
AIM> DEL WINDOW WINDOW_A
AIM> DEL WINDOW WINDOW_B
AIM> DEL WINDOW WINDOW_C
AIM> DEL IMAGE IMAGE_1
AIM> EXIT



___________________________BOTTOM-OF-SCREEN___________________________
```

Comments: IMAGE_1 has been deleted from the AIM's internal list of images.

Command: EXIT - Exit the AIM session.

AIM SUCCESSFULLY EXITED

```
_______________________________________________________________________________
 AIM              AIM       AIM CLI       Running                          AF
AIM> CREATE OBJECT_TYPE=>IMAGE IMAGE_NAME=>IMAGE_1
AIM> CREATE OBJECT_TYPE=>WINDOW WINDOW_NAME=>WINDOW_A
AIM> CREATE OBJECT_TYPE=>WINDOW WINDOW_B
AIM> CR WINDOW WINDOW_C
AIM> ASSOC WINDOW_A IMAGE_1 1 8
AIM> ASSOC WINDOW_B IMAGE_1 9 8
AIM> ASSOC WINDOW_C IMAGE_1 17 8
AIM> GOTO IMAGE IMAGE_1
AIM> GOTO WINDOW WINDOW_C
AIM> DIS WINDOW_A IMAGE_1
AIM> DIS WINDOW_B IMAGE_1
AIM> DIS WINDOW_C IMAGE_1
AIM> DEL WINDOW WINDOW_A
AIM> DEL WINDOW WINDOW_B
AIM> DEL WINDOW WINDOW_C
AIM> DEL IMAGE IMAGE_1
AIM> EXIT
-)



___________________________BOTTOM-OF-SCREEN________________________________
```

Comments: The AIM has terminated its execution and the prompt from the underlying APSE reappears.

Command: None.

A.2 SESSION 2 - SCRIPT AND DEFINE COMMANDS

The following session demonstrates the use of the DEFINE TERMINAL, DEFINE BINDING, and SCRIPT commands. The DEFINE BINDING binds a keystroke(s) to a key sequence and DEFINE TERMINAL indicates to the AIM the type of terminal being used. The DEFINE's are shown first followed by the SCRIPT command. The actual keystrokes are demonstrated in the next session. The following script files are used in this session:

1. demo1.aim

```
CREATE  IMAGE  IMAGE_1
CREATE  IMAGE  IMAGE_2
CREATE  WINDOW  WINDOW_A
CREATE  WINDOW  WINDOW_B
CREATE  WINDOW  WINDOW_C
CREATE  WINDOW  WINDOW_D
ASSOC  WINDOW_A  IMAGE_1  1   8
ASSOC  WINDOW_B  IMAGE_1  9   8
ASSOC  WINDOW_C  IMAGE_1  17  8
ASSOC  WINDOW_C  IMAGE_2  1   12
ASSOC  WINDOW_D  IMAGE_2  13  12
```

INVOKE AIM FROM APSE

```
-) AIM
```

BOTTOM_OF_SCREEN

Comments:

Command: AIM - invoke the AIM from the APSE.

SWITCH TO AIM IMAGE

```
________________________________________________________________________
 MAIN            MAIN          FRCH:016        Running               AF

AOS/VS CLI REV 05.01.00.00       9-APR-85      13:26:43 {F10}














____________________________BOTTOM_OF_SCREEN____________________________
```

Comments: The AIM initializes the current image to be MAIN. To enter AIM textual commands, the user must communicate through the AIM window to the AIM command interpreter.

Command: {F10} - Set the current image to be the AIM image. This is the default binding for the SWITCH_TO_AIM keystroke.

DEFINE TERMINAL

```
 AIM                AIM      AIM CLI       Running                              AF
AIM> DEFINE TERMINAL "TV970"
















______________________________BOTTOM_OF_SCREEN_____________________________
```

Comments: The cursor is now at the top of the AIM image. AIM commands can only be entered in the AIM image. Control has been passed to the AIM CI executing in the AIM window.

Command: DEFINE TERMINAL "TV970" - Define the terminal to be a Televideo TV970.

DEFINE KEYSTROKES - CLEAR_WINDOW

```
 AIM                 AIM        AIM CLI          Running                        AF
AIM> DEFINE TERMINAL "TV970"
AIM> DEFINE BINDING CLEAR_WINDOW F2
```

```
_____________________________BOTTOM_OF_SCREEN___________________________
```

Comments: The terminal was defined as a TV970. Next each keystroke will be defined.

Command: DEFINE BINDING CLEAR_WINDOW F2 This defines the CLEAR_WINDOW keystroke to be the function key shown.

DEFINE KEYSTROKES - NEXT_IMAGE

```
 AIM             AIM      AIM CLI       Running                      AF
AIM> DEFINE TERMINAL "TV970"
AIM> DEFINE BINDING CLEAR_WINDOW  F2
AIM> DEFINE BINDING NEXT_IMAGE F3









______________________________BOTTOM_OF_SCREEN______________________________
```

Comments: The first one is the CLEAR_WINDOW keystroke. This keystroke, when used, will clear the current window except for the header information.

Command: DEFINE BINDING NEXT_IMAGE F3 This defines the NEXT_IMAGE keystroke to be the function key shown.

DEFINE KEYSTROKES - NEXT_VIEWPORT

```
 AIM              AIM       AIM CLI        Running                          AF
AIM> DEFINE TERMINAL "TV970"
AIM> DEFINE BINDING CLEAR_WINDOW  F2
AIM> DEFINE BINDING NEXT_IMAGE  F3
AIM> DEFINE BINDING NEXT_VIEWPORT F5













_______________________________BOTTOM_OF_SCREEN________________________________
```

Comments: The NEXT_IMAGE keystroke, when used, causes the cursor to move to the image following the current image. The move is based on the alphabetical ordering of the images.

Command: DEFINE BINDING NEXT_VIEWPORT F5 This defines the NEXT_VIEWPORT keystroke to be the function key shown.

DEFINE KEYSTROKES - PREVIOUS_IMAGE

```
 AIM              AIM       AIM CLI        Running                          AF
AIM> DEFINE TERMINAL "TV970"
AIM> DEFINE BINDING CLEAR_WINDOW  F2
AIM> DEFINE BINDING NEXT_IMAGE  F3
AIM> DEFINE BINDING NEXT_VIEWPORT  F5
AIM> DEFINE BINDING PREVIOUS_IMAGE F6









________________________________BOTTOM_OF_SCREEN________________________________
```

Comments: The NEXT_VIEWPORT keystroke causes the cursor to move to the next viewport of the current image. The move is down the screen to the next viewport unless the cursor is in the last viewport of the screen. In this case, the cursor moves to the viewport at the top of the screen.

Command: DEFINE BINDING PREVIOUS_IMAGE F6 This defines the PREVIOUS_IMAGE keystroke to be the function key shown.

DEFINE KEYSTROKES - PREVIOUS_VIEWPORT

```
 AIM            AIM       AIM CLI       Running                         AF
AIM> DEFINE TERMINAL "TV970"
AIM> DEFINE BINDING CLEAR_WINDOW  F2
AIM> DEFINE BINDING NEXT_IMAGE  F3
AIM> DEFINE BINDING NEXT_VIEWPORT  F5
AIM> DEFINE BINDING PREVIOUS_IMAGE  F6
AIM> DEFINE BINDING PREVIOUS_VIEWPORT F7












_______________________________BOTTOM_OF_SCREEN_______________________________
```

Comments: The PREVIOUS_IMAGE keystroke is the complement to the NEXT_IMAGE keystroke. The cursor will move to the previous image based on the alphabetical ordering of the images.

Command: DEFINE BINDING PREVIOUS_VIEWPORT F7 This defines the PREVIOUS_VIEWPORT keystroke to be the function key shown.

DEFINE KEYSTROKES - REDISPLAY_SCREEN

```
 AIM              AIM        AIM CLI        Running                                  AF
AIM> DEFINE TERMINAL "TV970"
AIM> DEFINE BINDING CLEAR_WINDOW  F2
AIM> DEFINE BINDING NEXT_IMAGE  F3
AIM> DEFINE BINDING NEXT_VIEWPORT  F5
AIM> DEFINE BINDING PREVIOUS_IMAGE  F6
AIM> DEFINE BINDING PREVIOUS_VIEWPORT  F7
AIM> DEFINE BINDING REDISPLAY_SCREEN F8









______________________________BOTTOM_OF_SCREEN______________________________
```

Comments: The PREVIOUS_VIEWPORT keystroke is the complement of the NEXT_VIEWPORT keystroke. It moves the cursor to the viewport above the current viewport on the screen. If there is no viewport above the current viewport then the cursor moves to the last viewport on the screen.

Command: DEFINE BINDING REDISPLAY_SCREEN F8 This defines the REDISPLAY_SCREEN keystroke to be the function key shown.

DEFINE KEYSTROKES - RETURN_TO_PREVIOUS_IMAGE

```
 AIM                AIM       AIM CLI         Running                         AF
AIM> DEFINE TERMINAL "TV970"
AIM> DEFINE BINDING CLEAR_WINDOW  F2
AIM> DEFINE BINDING NEXT_IMAGE  F3
AIM> DEFINE BINDING NEXT_VIEWPORT  F5
AIM> DEFINE BINDING PREVIOUS_IMAGE  F6
AIM> DEFINE BINDING PREVIOUS_VIEWPORT  F7
AIM> DEFINE BINDING REDISPLAY_SCREEN  F8
AIM> DEFINE BINDING RETURN_TO_PREVIOUS_IMAGE F9










________________________________BOTTOM_OF_SCREEN_________________________________
```

Comments: The REDISPLAY_SCREEN keystroke repaints the entire screen. Any information that has been written on the screen by the operating system will not be REDISPLAYed.

Command: DEFINE BINDING RETURN_TO_PREVIOUS_IMAGE F9 This defines the RETURN_TO_PREVIOUS_IMAGE keystroke to be the function key shown.

DEFINE KEYSTROKES - SWITCH_TO_AIM

```
 AIM              AIM       AIM CLI        Running                              AF
AIM> DEFINE TERMINAL "TV970"
AIM> DEFINE BINDING CLEAR_WINDOW  F2
AIM> DEFINE BINDING NEXT_IMAGE  F3
AIM> DEFINE BINDING NEXT_VIEWPORT  F5
AIM> DEFINE BINDING PREVIOUS_IMAGE  F6
AIM> DEFINE BINDING PREVIOUS_VIEWPORT  F7
AIM> DEFINE BINDING REDISPLAY_SCREEN  F8
AIM> DEFINE BINDING RETURN_TO_PREVIOUS_IMAGE  F9
AIM> DEFINE BINDING SWITCH_TO_AIM F10









______________________________BOTTOM_OF_SCREEN______________________________
```

Comments: The RETURN_TO_PREVIOUS_IMAGE keystroke is different from the PREVIOUS_IMAGE keystroke in that the cursor movement associated with this keystroke is not based on the alphabetical ordering of the images. This keystroke moves the cursor to the image that the cursor was in prior to the current image.

Command: DEFINE BINDING SWITCH_TO_AIM F10 This defines the SWITCH_TO_AIM keystroke to be the function key shown.

DEFINE KEYSTROKES - ABORT_SCRIPT

```
 AIM            AIM       AIM CLI        Running                            AF
AIM> DEFINE TERMINAL "TV970"
AIM> DEFINE BINDING CLEAR_WINDOW  F2
AIM> DEFINE BINDING NEXT_IMAGE  F3
AIM> DEFINE BINDING NEXT_VIEWPORT  F5
AIM> DEFINE BINDING PREVIOUS_IMAGE  F6
AIM> DEFINE BINDING PREVIOUS_VIEWPORT  F7
AIM> DEFINE BINDING REDISPLAY_SCREEN  F8
AIM> DEFINE BINDING RETURN_TO_PREVIOUS_IMAGE  F9
AIM> DEFINE BINDING SWITCH_TO_AIM  F10
AIM> DEFINE BINDING ABORT_SCRIPT F1













______________________________BOTTOM_OF_SCREEN______________________________
```

Comments: The SWITCH_TO_AIM keystroke returns the cursor to the AIM image. This keystroke is used to restart scripts that have stopped because of embedded INFO or HELP commands. Also, if a script contains a GOTO command, when this keystroke is entered the cursor will return to the AIM window and the script will then resume.

Command: DEFINE BINDING ABORT_SCRIPT F1 This defines the ABORT_SCRIPT keystroke to be the function key shown.

DEFINE KEYSTROKES - NEXT_PAGE

```
 AIM             AIM       AIM CLI       Running                      AF
AIM> DEFINE TERMINAL "TV970"
AIM> DEFINE BINDING CLEAR_WINDOW F2
AIM> DEFINE BINDING NEXT_IMAGE  F3
AIM> DEFINE BINDING NEXT_VIEWPORT  F5
AIM> DEFINE BINDING PREVIOUS_IMAGE  F6
AIM> DEFINE BINDING PREVIOUS_VIEWPORT  F7
AIM> DEFINE BINDING REDISPLAY_SCREEN  F8
AIM> DEFINE BINDING RETURN_TO_PREVIOUS_IMAGE  F9
AIM> DEFINE BINDING SWITCH_TO_AIM  F10
AIM> DEFINE BINDING ABORT_SCRIPT  F1
AIM> DEFINE BINDING NEXT_PAGE F4









_______________________________BOTTOM_OF_SCREEN_______________________________
```

Comments: The ABORT_SCRIPT keystroke aborts all scripts. This includes all levels of nested scripts.

Command: DEFINE BINDING NEXT_PAGE F4 This defines the NEXT_PAGE keystroke to be the function key shown.

EXECUTE THE SCRIPT COMMAND

```
 AIM              AIM      AIM CLI       Running                      AF
AIM> DEFINE TERMINAL "TV970"
AIM> DEFINE BINDING CLEAR_WINDOW  F2
AIM> DEFINE BINDING NEXT_IMAGE  F3
AIM> DEFINE BINDING NEXT_VIEWPORT  F5
AIM> DEFINE BINDING PREVIOUS_IMAGE  F6
AIM> DEFINE BINDING PREVIOUS_VIEWPORT  F7
AIM> DEFINE BINDING REDISPLAY_SCREEN  F8
AIM> DEFINE BINDING RETURN_TO_PREVIOUS_IMAGE  F9
AIM> DEFINE BINDING SWITCH_TO_AIM  F10
AIM> DEFINE BINDING ABORT_SCRIPT  F1
AIM> DEFINE BINDING NEXT_PAGE  F4
AIM> SCRIPT "demol.aim"









______________________________BOTTOM_OF_SCREEN____________________________
```

Comments: The NEXT_PAGE keystroke is used in conjunction with the SET FULL command (see session 4). When the SET FULL flag is on, a flag is set in the header indicating that there is more information than will fit in the window. If the next page of information is to be seen, the NEXT_PAGE keystroke is used.

Command: SCRIPT "demol.aim" - Invoke a script called demol. (demol is a system dependent name.)

ECHO THE SCRIPT COMMANDS

```
 AIM            AIM       AIM CLI       Running                   AF
AIM> DEFINE BINDING CLEAR_WINDOW  F2
AIM> DEFINE BINDING NEXT_IMAGE  F3
AIM> DEFINE BINDING NEXT_VIEWPORT  F5
AIM> DEFINE BINDING PREVIOUS_IMAGE  F6
AIM> DEFINE BINDING PREVIOUS_VIEWPORT  F7
AIM> DEFINE BINDING REDISPLAY_SCREEN  F8
AIM> DEFINE BINDING RETURN_TO_PREVIOUS_IMAGE  F9
AIM> DEFINE BINDING SWITCH_TO_AIM  F10
AIM> DEFINE BINDING ABORT_SCRIPT  F1
AIM> DEFINE BINDING NEXT_PAGE  F2
AIM> SCRIPT "demol.aim"
AIM> CREATE  IMAGE  IMAGE_1
AIM> CREATE  IMAGE  IMAGE_2
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> CREATE  WINDOW  WINDOW_D
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> ASSOC  WINDOW_C  IMAGE_2  1   12
AIM> ASSOC  WINDOW_D  IMAGE_2  13  12
AIM> EXIT
_______________________________BOTTOM_OF_SCREEN______________________________
```

Comments: The SCRIPT command invokes a previously created command script. The script contains AIM commands and comments only. The script can be created using a system supplied editor. Nesting of script is allowed. See Session 3 for an example of nested scripts. The SCRIPT is echoed to the screen as each command is read and executed by the AIM. This completes the session on the DEFINEs and SCRIPT. Exit the AIM.

Command: EXIT - Exit this session.

ECHO THE SCRIPT COMMANDS

```
 AIM              AIM       AIM CLI       Running                          AF
AIM> DEFINE BINDING NEXT_IMAGE  F3
AIM> DEFINE BINDING NEXT_VIEWPORT  F5
AIM> DEFINE BINDING PREVIOUS_IMAGE  F6
AIM> DEFINE BINDING PREVIOUS_VIEWPORT  F7
AIM> DEFINE BINDING REDISPLAY_SCREEN  F8
AIM> DEFINE BINDING RETURN_TO_PREVIOUS_IMAGE  F9
AIM> DEFINE BINDING SWITCH_TO_AIM  F10
AIM> DEFINE BINDING ABORT_SCRIPT  F1
AIM> DEFINE BINDING NEXT_PAGE  F4
AIM> SCRIPT "demo1.aim"
AIM> CREATE  IMAGE  IMAGE_1
AIM> CREATE  IMAGE  IMAGE_2
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> CREATE  WINDOW  WINDOW_D
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> ASSOC  WINDOW_C  IMAGE_2  1   12
AIM> ASSOC  WINDOW_D  IMAGE_2  13  12
AIM> EXIT
-)
______________________________BOTTOM_OF_SCREEN______________________________
```

Comments: AIM exited.

Command: None.

A.3 SESSION 3 - KEYSTROKE COMMANDS

The following session demonstrates the use of the KEYSTROKE commands. The session is started by executing a script that defines the keystrokes and terminal type (see Session 2). The following text files are used in this session:

1. demo1.aim

```
    CREATE  IMAGE   IMAGE_1
    CREATE  IMAGE   IMAGE_2
    CREATE  WINDOW  WINDOW_A
    CREATE  WINDOW  WINDOW_B
    CREATE  WINDOW  WINDOW_C
    CREATE  WINDOW  WINDOW_D
    ASSOC  WINDOW_A  IMAGE_1  1   8
    ASSOC  WINDOW_B  IMAGE_1  9   8
    ASSOC  WINDOW_C  IMAGE_1  17  8
    ASSOC  WINDOW_C  IMAGE_2  1   12
    ASSOC  WINDOW_D  IMAGE_2  13  12
```

2. demo13.aim

```
    SCRIPT "demo1.aim"
    SCRIPT "demo14.aim"
```

3. demo14.aim

```
    DEFINE  BINDING  CLEAR_WINDOW  F2
    DEFINE  BINDING  NEXT_IMAGE  F3
    DEFINE  BINDING  NEXT_VIEWPORT  F5
    DEFINE  BINDING  PREVIOUS_IMAGE  F6
    DEFINE  BINDING  PREVIOUS_VIEWPORT  F7
    DEFINE  BINDING  REDISPLAY-SCREEN  F8
    DEFINE  BINDING  RETURN_TO_PREVIOUS_IMAGE  F9
    DEFINE  BINDING  SWITCH_TO_AIM  F10
    DEFINE  BINDING  ABORT_SCRIPT  F1
    DEFINE  BINDING  NEXT_PAGE  F4
    INFO  KEYS *
    GOTO  IMAGE  IMAGE_1
```

4. demo12.aim

   <This script contains comments only>

INVOKE AIM FROM APSE

```
-) AIM















________________________________BOTTOM-OF-SCREEN________________________________
```

Comments:

Command: AIM - invoke the AIM from the APSE.

SWITCH TO AIM IMAGE

```
________________________________________________________________________
MAIN           MAIN         FRCH:016       Running                    AF

AOS/VS CLI REV 05.01.00.00        9-APR-85      13:26:43 {F10}




















_______________________________BOTTOM-OF-SCREEN_________________________
```

Comments: The AIM initializes the current image to be MAIN. To enter AIM textual commands, the user must communicate through the AIM image to the AIM command interpreter.

Command: {F10} - Set the current image to be the AIM image. This keystroke is the default for Switch_to_AIM.

INVOKE SCRIPT

```
________________________________________________________________________________
 AIM              AIM       AIM CLI        Running                          AF
AIM> SCRIPT "demo13.aim"














________________________________BOTTOM-OF-SCREEN________________________________
```

Comments: The cursor is now at the top of the AIM image. AIM commands ca only be entered in the AIM image. Control has been passed to the AIM C executing in the AIM window.

Command: SCRIPT "demo13.aim" - Invoke a script called demo13.ai (demo13.aim is a system dependent name.)

ECHO SCRIPT LINES

```
 AIM                AIM        AIM CLI          Running         More..                AF
AIM> SCRIPT "demol.aim"
AIM> CREATE  IMAGE  IMAGE_1
AIM> CREATE  IMAGE  IMAGE_2
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> CREATE  WINDOW  WINDOW_D
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> ASSOC  WINDOW_C  IMAGE_2  1   12
AIM> ASSOC  WINDOW_D  IMAGE_2  13  12
AIM> SCRIPT "demol4.aim"
AIM> DEFINE BINDING CLEAR_WINDOW  F2
AIM> DEFINE BINDING NEXT_IMAGE  F3
AIM> DEFINE BINDING NEXT_VIEWPORT  F5
AIM> DEFINE BINDING PREVIOUS_IMAGE  F6
AIM> DEFINE BINDING PREVIOUS_VIEWPORT  F7
AIM> DEFINE BINDING REDISPLAY_SCREEN  F8
AIM> DEFINE BINDING RETURN_TO_PREVIOUS_IMAGE  F9
AIM> DEFINE BINDING SWITCH_TO_AIM  F10
AIM> DEFINE BINDING ABORT_SCRIPT  F1
AIM> DEFINE BINDING NEXT_PAGE  F4 {F4}
_______________________________BOTTOM-OF-SCREEN_______________________________
```

Comments: Invoke a script. Script demo3 actually invokes script demol and then script demo4. The first script sets up an environment in which to test the keystrokes. The second script will define the keystrokes with the same key sequences as in Session 2. Note that the screen has scrolled. The initial command to invoke a script and the first command of that script (SCRIPT demol.aim) have scrolled off the screen.

Command: {F4}

INFO ON KEYSTROKES

```
 AIM              AIM       AIM CLI       Running                        AF
AIM> DEFINE BINDING PREVIOUS_IMAGE  F6
AIM> DEFINE BINDING PREVIOUS_VIEWPORT  F7
AIM> DEFINE BINDING REDISPLAY_SCREEN  F8
AIM> DEFINE BINDING RETURN_TO_PREVIOUS_IMAGE  F9
AIM> DEFINE BINDING SWITCH_TO_AIM  F10
AIM> DEFINE BINDING ABORT_SCRIPT  F1
AIM> DEFINE BINDING NEXT_PAGE  F4
AIM> INFO  KEYS *

Keystroke Command Name           Key Sequence

ABORT_SCRIPT                        F1
CLEAR_WINDOW                        F2
NEXT_IMAGE                          F3
NEXT_PAGE                           F4
NEXT_VIEWPORT                       F5
PREVIOUS_IMAGE                      F6
PREVIOUS_VIEWPORT                   F7
REDISPLAY_SCREEN                    F8
RETURN_TO_PREVIOUS_IMAGE            F9
SWITCH_TO_AIM                       F10

INFO Object?  {return}
______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: At this point, the INFO command is used to see that the assignments made for the keystrokes exist. The information on all the AIM keystrokes is displayed. Note that script I/O has been switched to interactive while in the INFO utility.

Command: {return} - Exit the current level of the INFO utility with a carriage return.

AD-A198151

NAVAL OCEAN SYSTEMS CENTER, SAN DIEGO, CA
APSE INTERACTIVE MONITOR – USER'S MANUAL-ADE™
VERSION BY: TEXAS INSTRUMENTS INC.

3 OF 4
NOSC TD 1218
UNCLASSIFIED
JAN 1988

UNCLASSIFIED

RESUME SCRIPT

```
_______________________________________________________________________________
 AIM             AIM      AIM CLI       Running                         AF
AIM> DEFINE BINDING REDISPLAY_SCREEN  F8
AIM> DEFINE BINDING RETURN_TO_PREVIOUS_IMAGE  F9
AIM> DEFINE BINDING SWITCH_TO_AIM  F10
AIM> DEFINE BINDING ABORT_SCRIPT  F1
AIM> DEFINE BINDING NEXT_PAGE  F4
AIM> INFO  KEYS *

Keystroke Command Name          Key Sequence

ABORT_SCRIPT                       F1
CLEAR_WINDOW                       F2
NEXT_IMAGE                         F3
NEXT_PAGE                          F4
NEXT_VIEWPORT                      F5
PREVIOUS_IMAGE                     F6
PREVIOUS_VIEWPORT                  F7
REDISPLAY_SCREEN                   F8
RETURN_TO_PREVIOUS_IMAGE           F9
SWITCH_TO_AIM                      F10

INFO Object?
AIM> GOTO IMAGE IMAGE_1
_______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The INFO utility has been exited. The script resumes at this point.

Command: GOTO IMAGE IMAGE_1 - This command is in the script. The GOTO command puts the cursor to the specified image.

SWITCH TO NEXT VIEWPORT

```
 IMAGE_1         WINDOW_A        FRCH:017        Running                    AF

AOS/VS CLI    REV 05.01.00.00        11-APR-85    8:55:29 {F5}



 IMAGE_1         WINDOW_B        FRCH:018        Running                    AF

AOS/VS CLI    REV 05.01.00.00        11-APR-85    8:55:29



 IMAGE_1         WINDOW_C        FRCH:019         Running                    AF

AOS/VS CLI    REV 05.01.00.00        11-APR-85    8:55:29



-- ----------------------------BOTTOM-OF-SCREEN------------------------------
```

Comments: The next script command is a GOTO. Go to IMAGE_1. IMAGE_1 was previously created in the first script, demo1. A demonstration of most of the keystrokes will be done once out of the AIM image.

Command: {F5} - NEXT_VIEWPORT keystroke is entered.

```
                     SWITCH TO PREVIOUS VIEWPORT
____________________________________________________________________
 IMAGE_1        WINDOW_A       [illegible]       Running             AF

AOS/VS CLI   REV 05.01.00.00     [illegible]     8:55:29



 IMAGE_1        WINDOW_B       FRCH:018         Running             AF

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29 {F7}



 IMAGE_1        WINDOW_C       FRCH:019          Running             AF

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29



_____________________________BOTTOM-OF-SCREEN_______________________
```

Comments: The first keystroke is NEXT_VIEWPORT. The cursor will move to viewport IMAGE_1, WINDOW_B.

Command: {F7} - PREVIOUS_VIEWPORT keystroke. Move the cursor back to IMAGE_1, WINDOW_A.

REDISPLAY SCREEN

```
 IMAGE_1        WINDOW_A       FRCH:017       Running                      AF

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29
) *** [OPER] System going down in five minutes *** {F8}



 IMAGE_1        WINDOW_B       FRCH:018       Running                      AF

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29



 IMAGE_1        WINDOW_C       FRCH:019        Running                     AF

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29



____________________________BOTTOM-OF-SCREEN________________________________
```

Comments: Now the cursor has moved back to viewport IMAGE_1, WINDOW_A.

Command: {F8} - REDISPLAY_SCREEN keystroke. This keystroke repaints the screen.

SWITCH TO NEXT ALPHA IMAGE

```
 IMAGE_1        WINDOW_A       FRCH:017       Running                   AF

AOS/VS CLI   REV 05.01.00.00      11-APR-85    8:55:29 {F3}



 IMAGE_1        WINDOW_B       FRCH:018       Running                   AF

AOS/VS CLI   REV 05.01.00.00      11-APR-85    8:55:29



 IMAGE_1        WINDOW_C       FRCH:019        Running                   AF

AOS/VS CLI   REV 05.01.00.00      11-APR-85    8:55:29



____________________________BOTTOM-OF-SCREEN____________________________
```

Comments: If the screen is written on by the system it can be redisplayed with the REDISPLAY_SCREEN keystroke.

Command: {F3} - NEXT_IMAGE keystroke. Move the cursor to the next image.

SWITCH TO AIM

```
 IMAGE_2          WINDOW_C        FRCH:019         Running                       AF

AOS/VS CLI    REV 05.01.00.00       11-APR-85     8:55:29 {F10}




 IMAGE_2          WINDOW_D         FRCH:020         Running                      AF

AOS/VS CLI REV 05.01.00.00           11-APR-85     8:55:29




____________________________BOTTOM-OF-SCREEN____________________________
```

Comments: To demonstrate the CLEAR_WINDOW keystroke first enter some text in the window. After entering the text clear the window.

Command: {F10} - Enter the SWITCH_TO_AIM keystroke to return to the AIM window.

SWITCH TO PREVIOUS IMAGE

```
 AIM              AIM        AIM CLI        Running                        AF
AIM> DEFINE BINDING SWITCH_TO_AIM  F10
AIM> DEFINE BINDING ABORT_SCRIPT  F1
AIM> DEFINE BINDING NEXT_PAGE  F4
AIM> INFO  KEYS *

Keystroke Command Name          Key Sequence

ABORT_SCRIPT                       F1
CLEAR_WINDOW                       F2
NEXT_IMAGE                         F3
NEXT_PAGE                          F4
NEXT_VIEWPORT                      F5
PREVIOUS_IMAGE                     F6
PREVIOUS_VIEWPORT                  F7
REDISPLAY_SCREEN                   F8
RETURN_TO_PREVIOUS_IMAGE           F9
SWITCH_TO_AIM                      F10

INFO Object?

AIM> GOTO  IMAGE  IMAGE_1
AIM> {F9}
________________________________BOTTOM-OF-SCREEN________________________________
```

Comments: The SWITCH_TO_AIM keystroke will move the cursor back to the AIM image.

Command: {F9} - RETURN_TO_PREVIOUS_IMAGE keystroke. Return to IMAGE_2.

SWITCH TO PREVIOUS ALPHA IMAGE

```
 IMAGE_2         WINDOW_C       FRCH:019       Suspended                    AFS

AOS/VS CLI    REV 05.01.00.00        11-APR-85     8:55:29 {F6}






 IMAGE_2         WINDOW_D        FRCH:020       Suspended                   AFS

AOS/VS CLI  REV 05.01.00.00           11-APR-85     8:55:29





______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: Note that the script has completed and that commands are entered interactively. The RETURN_TO_PREVIOUS_IMAGE keystroke returns the cursor to IMAGE_2. Note that the suspended flags are set. This resulted from executing the SWITCH_TO_AIM keystroke.

Command: {F6} - PREVIOUS_IMAGE keystroke. This keystroke will put the cursor back in IMAGE_1.

SWITCH TO AIM

```
 IMAGE_1          WINDOW_A        FRCH:017        Suspended                    AFS

AOS/VS CLI    REV 05.01.00.00        11-APR-85    8:55:29 {F10}



 IMAGE_1          WINDOW_B        FRCH:018        Suspended                    AFS

AOS/VS CLI    REV 05.01.00.00        11-APR-85    8:55:29



 IMAGE_1          WINDOW_C        FRCH:019        Suspended                    AFS

AOS/VS CLI    REV 05.01.00.00        11-APR-85    8:55:29



______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: Go back to IMAGE_1 by entering the PREVIOUS_IMAGE keystroke to move the cursor to the previous image based on alphabetical order. Enter that keystroke now.

Command: {F10} - SWITCH_TO_AIM keystroke. Go back to the AIM image.

SCRIPT RESTARTS - ABORT IT

```
 AIM              AIM       AIM CLI        Running                        AF
AIM> DEFINE BINDING SWITCH_TO_AIM  F10
AIM> DEFINE BINDING ABORT_SCRIPT  F1
AIM> DEFINE BINDING NEXT_PAGE  F4
AIM> INFO  KEYS *

Keystroke Command Name          Key Sequence

ABORT_SCRIPT                       F1
CLEAR_WINDOW                       F2
NEXT_IMAGE                         F3
NEXT_PAGE                          F4
NEXT_VIEWPORT                      F5
PREVIOUS_IMAGE                     F6
PREVIOUS_VIEWPORT                  F7
REDISPLAY_SCREEN                   F8
RETURN_TO_PREVIOUS_IMAGE           F9
SWITCH_TO_AIM                      F10

INFO Object? {return}

AIM> GOTO  IMAGE  IMAGE_1
AIM> {F2}
______________________________BOTTOM-OF-SCREEN________________________
```

Comments: The cursor has now moved back to the AIM image. The user ca
now enter AIM commands.

Command: {F2} Clear window

```
 AIM                AIM         AIM CLI        Running                                  AF
AIM> SCRIPT "demol2.aim"









______________________________________BOTTOM-OF-SCREEN_____________________________________
```

Comments: The winow is cleared

Command: SCRIPT "demol2.aim" - Execute another script.

SCRIPT RESTARTS - ABORT IT

```
 AIM                 AIM       AIM CLI        Running                          AF
AIM> SCRIPT "demo12.aim"
{F1}
```

BOTTOM-OF-SCREEN

Comments: The script resumes when returning to the AIM image.

Command: {F1} - Abort the script before it gets started.

ENTER EXIT COMMAND

```
 AIM              AIM       AIM CLI       Running                         AF
AIM> SCRIPT "demo12.aim"
AIM>
Aborted execution of script file: :USER1:ATB:AIM:TESTING:DEMO12.AIM
AIM> EXIT















_________________________BOTTOM-OF-SCREEN______________________________
```

Comments: Use the ABORT_SCRIPT keystroke to abort the script before it starts running.

Command: EXIT - Exit the AIM.

EXIT

```
 AIM              AIM        AIM CLI        Running                         AF
AIM> SCRIPT "demo12.aim"
AIM>
Aborted execution of script file: :USER1:ATB:AIM:TESTING:DEMO12.AIM
AIM> EXIT
-)
```

BOTTOM-OF-SCREEN

Comments: This session is over so enter the exit command. The only keystroke not demonstrated was the NEXT_PAGE keystroke. It will be demonstrated in Session 4.

Command: None.

A.4 SESSION 4 - WINDOW RELATED COMMANDS

The following AIM command session demonstrates the use of the window related AIM commands, namely, SET and RESET. The following text files are used in this session:

1. demo11.aim

```
CREATE  IMAGE  IMAGE_1
CREATE  WINDOW  WINDOW_A
CREATE  WINDOW  WINDOW_B
CREATE  WINDOW  WINDOW_C
ASSOC  WINDOW_A  IMAGE_1  1   8
ASSOC  WINDOW_B  IMAGE_1  9   8
ASSOC  WINDOW_C  IMAGE_1  17  8
RESET AIM_SUSPENDS
```

2. jabberwocky

```
              JABBERWOCKY       by Lewis Carol

'Twas brillig, and the slithy toves
  Did gyre and gimble in the wabe:
All mimsy were the borogoves,
    And the mome raths outgrabe.

"Beware the Jabberwock, my son!
  The jaws that bite, the claws that catch!
Beware the Jubjub bird, and shun
    The frumious Bandersnatch!"

He took his vorpal sword in hand:
 Long time the manxome foe he sought--
So rested he by the Tumtum tree,
    And stood awhile in thought.

And, as in uffish thought he stood,
 The Jabberwock, with eyes of flame,
Came whiffling through the tulgey wood,
    And burbled as it came!

One, two! One, two! And through and through
 The vorpal blade went snicker-snack!
He left it dead, and with its head
    He went galumphing back.

"And has thou slain the Jabberwock?
 Come to my arms, my beamish boy!
O frabjous day! Callooh! Callay!"
    He chortled in his joy.
```

```
'Twas brillig, and the slithy toves
 Did gyre and gimble in the wabe:
All mimsy were the borogoves,
    And the mome raths outgrabe.
```

3. mabell

"The Day Bell System Died" by Lauren Weinstein

Long, long, time ago,
I can still remember,
When the local calls were "free".
And I knew if I paid my bill,
And never wished them any ill,
That the phone company would let me be...

But Uncle Sam said he knew better,
Split 'em up, for all and ever!
We'll foster competition:
It's good capital-ism!

I can't remember if I cried,
When my phone bill first tripled in size.
But something touched me deep inside,
The day... Bell System... died.

And we were singing...

Bye, bye, Ma Bell, why did you die?
We get static from Sprint and echo from MCI,
"Our local calls have us in hock!" we all cry.
Oh Ma Bell why did you have to die?
Ma Bell why did you have to die?

Is your office Step by Step,
Or have you gotten some Crossbar yet?
Everybody used to ask...
Oh, is TSPS coming soon?
IDDD will be a boon!
And, I hope to get a Touch-Tone phone, real soon...

The color phones are really neat,
And direct dialing can't be beat!
My area code is "low":
The prestige way to go!

Oh, they just raised phone booths to a dime!
Well, I suppose it's about time.
I remember how the payphones chimed,

The day... Bell System... died.

And we were singing...

Bye, bye, Ma Bell, why did you die?
We get static from Sprint and echo from MCI,
"Our local calls have us in hock!" we all cry.
Oh Ma Bell why did you have to die?
Ma Bell why did you have to die?

Back then we were all at one rate,
Phone installs didn't cause debate,
About who'd put which wire where...
Installers came right out to you,
No "phone stores" with their ballyhoo,
And 411 was free, seemed very fair!

But FCC wanted it seems,
To let others skim long-distance creams,
No matter 'bout the locals,
They're mostly all just yokels!

And so one day it came to pass,

That the great Bell System did collapse,
In rubble now, we all do mass,
The day... Bell System... died.

So bye, bye, Ma Bell, why did you die?
We get static from Sprint and echo from MCI,
"Our local calls have us in hock!" we all cry.
Oh Ma Bell why did you have to die?
Ma Bell why did you have to die?

I drove on out to Murray Hill,
To see Bell Labs, some time to kill,
But the sign there said the Labs were gone.
I went back to my old CO,
Where I'd had my phcne lines, years ago,
But it was empty, dark, and ever so forlorn...

No relays pulsed,
No data crooned,
No MF tones did play their tunes,
There wasn't a word spoken,
All carrier paths were broken...

And so that's how it all occurred,
Microwave horns just nests for birds,
Everything became so absurd,

The day... Bell System... died.

So bye, bye, Ma Bell, why did you die?
We get static from Sprint and echo from MCI,
"Our local calls have us in hock!" we all cry.
Oh Ma Bell why did you have to die?
Ma Bell why did you have to die?

We were singing:

Bye, bye, Ma Bell, why did you die?
We get static from Sprint and echo from MCI,
"Our local calls have us in hock!" we all cry.

Oh Ma Bell why did you have to die?

4. adalrm

Foreword

Ada is the result of a collective effort to design a common language for programming large scale and real-time systems.

The common high order language program began in 1974. The requirements of the United States Department of Defense were formalized in a series of documents which were extensively reviewed by the Services, industrial organizations, universities, and foreign military departments. The Ada language was designed in accordance with the final (1978) form of these requirements, embodied in the Steelman specification.

The Ada design team was led by Jean D. Ichbiah and has included Bernd Krieg-Brueckner, Brian A. Wichmann, Henry F. Ledgard, Jean-Claude Heliard, Jean-Loup Gailly, Jean-Raymond Abrial, John G.P. Barnes, Mike Woodger, Olivier Roubine, Paul N. Hilfinger, and Robert Firth.

At various stages of the project, several people closely associated with the design team made major contributions. They include J.B. Goodenough, R.F. Brender, M.W. Davis, G. Ferran, K. Lester, L. MacLaren, E. Morel, I.R. Nassi, I.C. Pyle, S.A. Schuman, and S.C. Vestal.

Two parallel efforts that were started in the second phase of this design had a deep influence on the language. One was the development of a formal definition using denotational semantics, with the participation of V. Donzeau-Gouge, G. Kahn, and B. Lang. The other was the design of a test translator with the participation of K. Ripken, P. Boullier, P. Cadiou, J. Holden, J.F. Hueras, R.G. Lange, and D.T. Cornhill. The entire effort

benefitted from the dedicated assistance of Lyn Churchill and Marion Myers, and the effective technical support of B. Gravem, W.L. Heimerdinger, and P. Cleve. H.G. Schmitz served as program manager.

Over the five years spent on this project, several intense week-long design reviews were conducted, with the participation of P. Belmont, B. Brosgol, P. Cohen, R. Dewar, A. Evans, G. Fisher, H. Harte, A.L. Hisgen, P. Knueven, M. Kronental, N. Lomuto, E. Ploedereder, G. Seegmueller, V. Stenning, D. Taffs, and also F. Belz, R. Converse, K. Correll, A.N. Habermann, J. Sammet, S. Squires, J. Teller, P. Wegner, and P.R. Wetherall.

INVOKE AIM FROM APSE

```
-) AIM









____________________________BOTTOM-OF-SCREEN____________________________
```

Comments:

Command: AIM - invoke the AIM from the APSE.

SWITCH TO AIM IMAGE

```
_____________________________________________________________________________
 MAIN           MAIN       FRCH:016       Running                         AF

AOS/VS CLI REV 05.01.00.00        9-APR-85     13:26:43 {F10}
























__________________________BOTTOM-OF-SCREEN___________________________________
```

Comments: The AIM initializes the current image to be MAIN. To enter AIM textual commands the user must communicate through the AIM image to the AIM command interpreter.

Command: {F10} - Set the current image to be the AIM image.

INVOKE SCRIPT

```
 AIM                AIM       AIM CLI       Running                                AF
AIM> SCRIPT "demoll.aim"















_______________________________BOTTOM-OF-SCREEN_______________________________
```

Comments: Control has been passed to the AIM command interpreter in the AIM window.

Command: SCRIPT "demoll.aim" - Invoke a script. The demoll.aim script initializes the AIM environment to be used in this AIM command session.

ECHO SCRIPT LINES

```
 AIM              AIM      AIM CLI       Running                        F
AIM> SCRIPT "demo11.aim"
AIM> CREATE  IMAGE  IMAGE_1
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> RESET  AIM_SUSPENDS
AIM> GOTO IMAGE IMAGE_1















______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The command lines of the script file are individually echoed back to the console as they are read and interpreted.

Command: GOTO IMAGE IMAGE_1 Position the cursor in the top viewport of IMAGE_1.

```
                      CURRENT IMAGE ON SCREEN IS IMAGE_1
______________________________________________________________________________
 IMAGE_1         WINDOW_A        FRCH:017        Running                     F

AOS/VS CLI    REV 05.01.00.00       11-APR-85    8:55:29 {F10}



 IMAGE_1         WINDOW_B        FRCH:018        Running                     F

AOS/VS CLI    REV 05.01.00.00       11-APR-85    8:55:29



 IMAGE_1         WINDOW_C        FRCH:019         Running                    F

AOS/VS CLI    REV 05.01.00.00       11-APR-85    8:55:29



_____________________________BOTTOM-OF-SCREEN_________________________________
```

Comments: The cursor was positioned in the first viewport of IMAGE_1. Each window is executing its own APSE process, namely the APSE command interpreter; thus, the APSE command interpreter prompt, ")", appears in WINDOW_A, WINDOW_B, and WINDOW_C. Note that each viewport header displays the default flag settings for the windows ("AF").

Command: {F10} Switch control back to the AIM command interpreter in the AIM window.

RESET FULL COMMAND

```
 AIM            AIM      AIM CLI       Running                         F
AIM> SCRIPT "demo11.aim"
AIM> CREATE  IMAGE  IMAGE_1
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> RESET AIM_SUSPENDS
AIM> GOTO IMAGE IMAGE_1
AIM> RESET FULL WINDOW_B







________________________________BOTTOM-OF-SCREEN________________________________
```

Comments: The AIM window flag settings have changed due to the RESET AIM command. For every window, the default value of the SUSPENDS_OUTPUT_WHEN_FULL flag is TRUE; thus, by default, the associated APSE program output will be suspended when an entire window of data has been generated.

Command: RESET FULL WINDOW_B - Turn off the SUSPENDS_OUTPUT_WHEN_FULL flag for WINDOW_B allowing continuous scrolling of the output generated by the APSE program associated with WINDOW_B.

SET PADS FOR AIM WINDOW

```
 AIM              AIM       AIM CLI          Running                          F
AIM> SCRIPT "demo11.aim"
AIM> CREATE  IMAGE  IMAGE_1
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> RESET AIM_SUSPENDS
AIM> GOTO IMAGE IMAGE_1
AIM> RESET FULL WINDOW_B
AIM> SET PADS AIM








______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The FULL flag has been set for WINDOW_B.

Command: SET PADS AIM - Activate both an input and output pad file for logging the activity of the AIM window.

SET INPUT_PAD FOR WINDOW_A

```
 AIM            AIM      AIM CLI       Running                            FIO
AIM> SCRIPT "demo11.aim"
AIM> CREATE  IMAGE  IMAGE_1
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> RESET AIM_SUSPENDS
AIM> GOTO IMAGE IMAGE_1
AIM> RESET FULL WINDOW_B
AIM> SET PADS AIM
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:AIM1.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:AIM1.OUT
AIM> SET INPUT_PAD WINDOW_A




______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The SET PADS AIM command is now complete as both the input and output pad file names have been supplied by the AIM. Note, the AIM window flag settings have changed to "FIO" indicating to the user that both an input and output pad file are currently associated with the AIM window.

Command: SET INPUT_PAD WINDOW_A - Activate a pad file for logging the input activity of WINDOW_A.

SET OUTPUT_PAD FOR WINDOW_B

```
 AIM          AIM     AIM CLI       Running                           FIO
AIM> SCRIPT "demoll.aim"
AIM> CREATE  IMAGE   IMAGE_1
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> RESET AIM_SUSPENDS
AIM> GOTO IMAGE IMAGE_1
AIM> RESET FULL WINDOW_B
AIM> SET PADS AIM
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:AIM1.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:AIM2.OUT
AIM> SET INPUT_PAD WINDOW_A
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A3.INP
AIM> SET OUTPUT_PAD WINDOW_B




_______________________________BOTTOM-OF-SCREEN________________________________
```

Comments: The SET INPUT_PAD command is now complete as the AIM supplied a file name for WINDOW_A's input pad.

Command: SET OUTPUT_PAD WINDOW_B - Activate a pad file for logging the output activity of WINDOW_B.

SPECIFY OUTPUT PAD NAME

```
 AIM            AIM      AIM CLI       Running                          FIO
AIM> SCRIPT "demoll.aim"
AIM> CREATE  IMAGE  IMAGE_1
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> RESET AIM_SUSPENDS
AIM> GOTO IMAGE IMAGE_1
AIM> RESET FULL WINDOW_B
AIM> SET PADS AIM
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:AIM1.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:AIM2.OUT
AIM> SET INPUT_PAD WINDOW_A
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A3.INP
AIM> SET OUTPUT_PAD WINDOW_B
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOWB_4.OUT
AIM> SET PADS WINDOW_C




______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The AIM is providing the specification of an output pad file name for WINDOW_B.

Command: SET PADS WINDOW_C

SET PADS FOR WINDOW_C

```
 AIM                AIM      AIM CLI        Running                       FIO
AIM> SCRIPT "demo11.aim"
AIM> CREATE  IMAGE  IMAGE_1
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> RESET AIM_SUSPENDS
AIM> GOTO IMAGE IMAGE_1
AIM> RESET FULL WINDOW_B
AIM> SET PADS AIM
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:AIM1.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:AIM2.OUT
AIM> SET INPUT_PAD WINDOW_A
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A3.INP
AIM> SET OUTPUT_PAD WINDOW_B
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_B4.OUT
AIM> SET PADS WINDOW_C
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C5.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C6.OUT
AIM> GOTO IMAGE IMAGE_1

______________________________BOTTOM-OF-SCREEN________________________________
```

Comments: The SET OUTPUT_PAD command is now complete as the AIM supplied a file name for WINDOW_C's pads.

Command: GOTO IMAGE IMAGE_1

LIST A FILE IN WINDOW_A

```
 IMAGE_1        WINDOW_A        FRCH:017        Running                     FI

AOS/VS CLI   REV 05.01.00.00        11-APR-85    8:55:29 TYPE JAB.AIM



 IMAGE_1        WINDOW_B        FRCH:018        Running                     O

AOS/VS CLI   REV 05.01.00.00        11-APR-85    8:55:29



 IMAGE_1        WINDOW_C        FRCH:019         Running                     FIO

AOS/VS CLI   REV 05.01.00.00        11-APR-85    8:55:29


_______________________________BOTTOM-OF-SCREEN_______________________________
```

Comments: The window flag settings have changed as a direct result of the RESET AIM, RESET FULL WINDOW_B, SET INPUT_PAD WINDOW_A, SET OUTPUT_PAD WINDOW_B, and SET PADS WINDOW_C commands. WINDOW_A ("FI") has the FULL flag set (default) and its input pad activated. WINDOW_B ("O") has the FULL flag turned off and its output pad activated. WINDOW_C ("FIO") has the FULL flag set (default) and both its input and output pads activated.

Command: TYPE JAB.AIM - List the contents of the file jab.aim in WINDOW_A.

SWITCH TO THE PRECEDING VIEWPORT ON IMAGE_1

```
____________________________________________________________________________
 IMAGE_1        WINDOW_A       FRCH:017       Running                   FI

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29 TYPE JAB.AIM
  TYPE JAB.AIM
                 JABBERWOCKY              by Lewis Carol

'Twas brillig, and the slithy toves {F7}
 Did qyre and gimble in the wabe:
 IMAGE_1        WINDOW_B       FRCH:018       Running                   FO

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29



 IMAGE_1        WINDOW_C       FRCH:C19        Running                   FIO

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29



__________ __________________BOTTOM-OF-SCREEN_______________________________
```

Comments: The listing of the "jabberwocky" file commences in WINDOW_A and then the user enters the PREVIOUS_VIEWPORT keystroke.

Command: {F7} - Switch to the preceding viewport on IMAGE_1.

LIST ANOTHER FILE IN WINDOW_C

```
 IMAGE_1        WINDOW_A       FRCH:017       Running      More..      FI
So rested he by the Tumtum tree,
    And stood awhile in thought.

And, as in uffish thought he stood,
 The Jabberwock, with eyes of flame,
Came wiffling through the tulgey wood,
    And burbled as it came!
 IMAGE_1        WINDOW_B       FRCH:018       Running                  O

AOS/VS CLI    REV 05.01.00.00       11-APR-85    8:55:29



 IMAGE_1        WINDOW_C       FRCH:019        Running                  FIO

AOS/VS CLI    REV 05.01.00.00       11-APR-85    8:55:29 TYPE LRM.AIM



_____________________________BOTTOM-OF-SCREEN_____________________________
```

Comments: The preceding viewport is the third one since the VIEWPORT_MANAGER considers an image a circular list of viewports. Note the last 7 lines of WINDOW_A are listed in the first viewport and the "More.." flag is turned on since there is more than one window full of text in the "jabberwocky" file. The cursor is positioned at the top of the viewport associated with WINDOW_C.

Command: TYPE LRM.AIM - List the contents of "adalrm" in WINDOW_C.

SWITCH TO PRECEDING VIEWPORT

```
_________________________________________________________________________________
 IMAGE_1         WINDOW_A       FRCH:017       Running        More..     FI
So rested he by the Tumtum tree,
    And stood awhile in thought.

And, as in uffish thought he stood,
 The Jabberwock, with eyes of flame,
Came wiffling through the tulgey wood,
    And burbled as it came!
 IMAGE_1         WINDOW_B       FRCH:018       Running                   O

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29




 IMAGE_1         WINDOW_C       FRCH:019        Running                   FIO

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29 TYPE LRM.AIM
  TYPE LRM.AIM
                               Foreword

Ada is the result of a collective effort to design a common language
programming large scale and real-time systems. {F7}
_____________________________BOTTOM-OF-SCREEN____________________________________
```

Comments: The listing of the "adalrm" file commences in WINDOW_C as the user enters the PREVIOUS_VIEWPORT keystroke.

Command: {F7} - Switch to the preceding viewport on IMAGE_1.

LIST A FILE IN WINDOW_B

```
 IMAGE_1        WINDOW_A       FRCH:017       Running        More..    FI
So rested he by the Tumtum tree,
    And stood awhile in thought.

And, as in uffish thought he stood,
 The Jabberwock, with eyes of flame,
Came wiffling through the tulgey wood,
    And burbled as it came!
 IMAGE_1        WINDOW_B       FRCH:018       Running                  O

AOS/VS CLI   REV 05.01.00.00      11-APR-85    8:55:29 TYPE MABELL.AIM
  TYPE MABELL.AIM
        "The Day Bell System Died"  by Lauren Weinstein

Long, long, time ago,
I can still remember,
 IMAGE_1        WINDOW_C       FRCH:019        Running        More..   FIO

The Ada design team was led by Jean D. Ichbiah and has included Bernd
Krieg-Brueckner, Brian A. Wichmann, Henry F. Ledgard, Jean-Claude Heliard,
Jean-Loup Gailly, Jean-raymond Abrial, John G.P. Barnes, Mike Woodger,
Olivier Roubine, Paul N. Hilfinger, and Robert Firth.

At various stages of the project, several people closely associated with
_____________________________BOTTOM-OF-SCREEN_____________________________
```

Comments: Note the last 7 lines of WINDOOW_C are listed in the third viewport and the "More.." flag is turned on since there is more than one window full of text in the "adalrm" file. The cursor is positioned at the top of the viewport associated with WINDOW_B.

Command: TYPE MABELL.AIM - List the contents of "mabell" in WINDOW_B.

SWITCH TO AIM WINDOW

```
_____________________________________________________________________________
 IMAGE_1        WINDOW_A       FRCH:017       Running        More..    FI
So rested he by the Tumtum tree,
    And stood awhile in thought.

And, as in uffish thought he stood,
 The Jabberwock, with eyes of flame,
Came wiffling through the tulgey wood,
    And burbled as it came!
 IMAGE_1        WINDOW_B       FRCH:018       Running                  O

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29 TYPE MABELL.AIM
  TYPE MABELL.AIM
       "The Day Bell System Died"  by Lauren Weinstein

Long, long, time ago,
I can still remember, {F10}
 IMAGE_1        WINDOW_C       FRCH:019        Running        More..   FIO

The Ada design team was led by Jean D. Ichbiah and has included Bernd
Krieg-Brueckner, Brian A. Wichmann, Henry F. Ledgard, Jean-Claude Heliard,
Jean-Loup Gailly, Jean-raymond Abrial, John G.P. Barnes, Mike Woodger,
Olivier Roubine, Paul N. Hilfinger, and Robert Firth.

At various stages of the project, several people closely associated with
____________________________BOTTOM-OF-SCREEN_______________________________
```

Comments: The listing of the "mabell" file is executing in WINDOW_B as the user enters the SWITCH_TO_AIM keystroke.

Command: {F10} - Switch the input context to the AIM window in order to communicate with the AIM command interpreter.

RESET INPUT_PAD WINDOW_A

```
_______________________________________________________________   ___________________
 AIM              AIM       AIM CLI       Running                               FIO
AIM> SCRIPT "demo11.aim"
AIM> CREATE  IMAGE  IMAGE_1
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> RESET AIM_SUSPENDS
AIM> GOTO IMAGE IMAGE_1
AIM> RESET FULL WINDOW_B
AIM> SET PADS AIM
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:AIM1.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:AIM2.OUT
AIM> SET INPUT_PAD WINDOW_A
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A3.INP
AIM> SET OUTPUT_PAD WINDOW_B
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_B4.OUT
AIM> SET PADS WINDOW_C
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C5.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C6.OUT
AIM> GOTO IMAGE IMAGE_1
AIM> RESET INPUT_PAD WINDOW_A
________________________________BOTTOM-OF-SCREEN________________________________
```

Comments: Control has returned to the AIM command interpreter in the AIM window.

Command: RESET INPUT_PAD WINDOW_A - Turn off the recording of WINDOW_A's input.

RESET OUTPUT_PAD WINDOW_B

```
 AIM            AIM     AIM CLI        Running                     FIO
AIM> CREATE IMAGE IMAGE_1
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> GOTO IMAGE IMAGE_1
AIM> RESET AIM_SUSPENDS
AIM> RESET FULL WINDOW_B
AIM> SET PADS AIM
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:AIM1.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:AIM2.OUT
AIM> SET INPUT_PAD WINDOW_A
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A3.INP
AIM> SET OUTPUT_PAD WINDOW_B
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A4.OUT
AIM> SET PADS WINDOW_C
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C5.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C6.OUT
AIM> GOTO IMAGE IMAGE_1
AIM> RESET INPUT_PAD WINDOW_A
AIM> RESET OUTPUT_PAD WINDOW_B
______________________________BOTTOM-OF-SCREEN____________________________
```

Comments: WINDOW_A's input pad has been deactivated.

Command: RESET OUTPUT_PAD WINDOW_B - Turn off the recording of WINDOW_B's output.

```
                                     More
____________________________________________________________________________
 AIM            AIM      AIM CLI        Running       More..            FIO
AIM> CREATE  IMAGE  IMAGE_1
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> RESET AIM_SUSPENDS
AIM> GOTO IMAGE IMAGE_1
AIM> RESET FULL WINDOW_B
AIM> SET PADS AIM
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:AIM1.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:AIM2.OUT
AIM> SET INPUT_PAD WINDOW_A
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A3.INP
AIM> SET OUTPUT_PAD WINDOW_B
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_B4.OUT
AIM> SET PADS WINDOW_C
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C5.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C6.OUT
AIM> GOTO IMAGE IMAGE_1
AIM> RESET INPUT_PAD WINDOW_A
AIM> RESET OUTPUT_PAD WINDOW_B {F4}
______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: More flag set

Command: {F4}

RESET PADS WINDOW_C

```
 AIM            AIM      AIM CLI        Running                       FIO
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> RESET AIM_SUSPENDS
AIM> GOTO IMAGE IMAGE_1
AIM> RESET FULL WINDOW_B
AIM> SET PADS AIM
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:AIM1.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:AIM2.OUT
AIM> SET INPUT_PAD WINDOW_A
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A3.INP
AIM> SET OUTPUT_PAD WINDOW_B
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A4.OUT
AIM> SET PADS WINDOW_C
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C5.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C6.OUT
AIM> GOTO IMAGE IMAGE_1
AIM> RESET INPUT_PAD WINDOW_A
AIM> RESET OUTPUT_PAD WINDOW_B
AIM> RESET PADS WINDOW_C
________________________________BOTTOM-OF-SCREEN______________________________
```

Comments: WINDOW_B's output pad has been deactivated.

Command: RESET PADS WINDOW_C - Turn off the recording of WINDOW_C's input and output.

SET FULL WINDOW_B

```
_____________________________________________________________________________
 AIM            AIM       AIM CLI       Running                          AF
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> RESET AIM_SUSPENDS
AIM> GOTO IMAGE IMAGE_1
AIM> RESET FULL WINDOW_B
AIM> SET PADS AIM
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:AIM1.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:AIM2.OUT
AIM> SET INPUT_PAD WINDOW_A
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A3.INP
AIM> SET OUTPUT_PAD WINDOW_B
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A4.OUT
AIM> SET PADS WINDOW_C
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C5.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C6.OUT
AIM> GOTO IMAGE IMAGE_1
AIM> RESET INPUT_PAD WINDOW_A
AIM> RESET OUTPUT_PAD WINDOW_B
AIM> RESET PADS WINDOW_C
AIM> SET FULL WINDOW_B
______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: WINDOW_C's input and output pads have been deactivated.

Command: SET FULL WINDOW_B - Turn off continuous scrolling of the output generated by the APSE program associated with WINDOW_B.

GOTO WINDOW_A

```
 AIM            AIM       AIM CLI       Running                        FIO
AIM> CREATE  WINDOW  WINDOW_C
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> RESET AIM_SUSPENDS
AIM> GOTO IMAGE IMAGE_1
AIM> RESET FULL WINDOW_B
AIM> SET PADS AIM
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:AIM1.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:AIM2.OUT
AIM> SET INPUT_PAD WINDOW_A
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A3.INP
AIM> SET OUTPUT_PAD WINDOW_B
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A4.OUT
AIM> SET PADS WINDOW_C
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C5.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C6.OUT
AIM> GOTO IMAGE IMAGE_1
AIM> RESET INPUT_PAD WINDOW_A
AIM> RESET OUTPUT_PAD WINDOW_B
AIM> RESET PADS WINDOW_C
AIM> SET FULL WINDOW_B
AIM> GOTO WINDOW WINDOW_A
_______________________________BOTTOM-OF-SCREEN_______________________________
```

Comments: WINDOW_B's APSE program's output will NOT scroll continuously; it will be suspended upon filling up WINDOW_B.

Command: GOTO WINDOW WINDOW_A - Switch the input context to WINDOW_A.

DISPLAY THE NEXT PAGE OF INFORMATION FOR WINDOW_A

```
 IMAGE_1        WINDOW_A       FRCH:017       Running       More..     F
So rested he by the Tumtum tree,
    And stood awhile in thought.

And, as in uffish thought he stood,
 The Jabberwock, with eyes of flame,
Came wiffling through the tulgey wood,
    And burbled as it came!{F4}
 IMAGE_1        WINDOW_B       FRCH:018       Running                  F

Bye, bye, Ma Bell, why did you die?
We get static from Sprint and echo from MCI,
"Our local calls have us in hock!" we all cry.

Oh Ma Bell why did you have to die?

 IMAGE_1        WINDOW_C       FRCH:019        Running      More..     F

The Ada design team was led by Jean D. Ichbiah and has included Bernd
Krieg-Brueckner, Brian A. Wichmann, Henry F. Ledgard, Jean-Claude Heliard,
Jean-Loup Gailly, Jean-raymond Abrial, John G.P. Barnes, Mike Woodger,
Olivier Roubine, Paul N. Hilfinger, and Robert Firth.

At various stages of the project, several people closely associated with
_____________________________BOTTOM-OF-SCREEN_____________________________
```

Comments: The listing of the "mabell" file has finished in WINDOW_B; the cursor is positioned at the end of WINDOW_A's viepwort.

Command: {F4} - Display the next window full of text available for WINDOW_A.

GOTO PRECEDING VIEWPORT ON IMAGE_1

```
 ____________________________________________________________________________
  IMAGE_1         WINDOW_A        FRCH:017        Running                   F
O frabjous day! Callooh! Callay!"
   He chortled in his joy.

'Twas brillig, and the slithy toves
 Did gyre and gimble in the wabe:
All mimsy were the borogoves,
   And the mome raths outgrabe. {F7}
  IMAGE_1         WINDOW_B        FRCH:018        Running                   F

Bye, bye, Ma Bell, why did you die?
We get static from Sprint and echo from MCI,
"Our local calls have us in hock!" we all cry.

Oh Ma Bell why did you have to die?

  IMAGE_1         WINDOW_C        FRCH:019         Running       More..     F

The Ada design team was led by Jean D. Ichbiah and has included Bernd
Krieg-Brueckner, Brian A. Wichmann, Henry F. Ledgard, Jean-Claude Heliard,
Jean-Loup Gailly, Jean-raymond Abrial, John G.P. Barnes, Mike Woodger,
Olivier Roubine, Paul N. Hilfinger, and Robert Firth.

At various stages of the project, several people closely associated with
_____________________________BOTTOM-OF-SCREEN________________________________
```

Comments: The listing of the "jabberwocky" file has finished, and the APSE command interpreter prompt is the last line of the viewport. Note the next page (window full) of output generated by the APSE LIST utility is displayed (continuously scrolled) in the viewport associated with WINDOW_A.

Command: {F7} - Switch to the preceding viewport on IMAGE_1.

DISPLAY THE NEXT PAGE OF INFORMATION FOR WINDOW_C

```
______________________________________________________________________________
 IMAGE_1        WINDOW_A       FRCH:017       Running                    F
    He chortled in his joy.

'Twas brillig, and the slithy toves
 Did gyre and gimble in the wabe:
All mimsy were the borogoves,

    And the mome raths outgrabe.
 IMAGE_1        WINDOW_B       FRCH:018       Running                    F

Bye, bye, Ma Bell, why did you die?
We get static from Sprint and echo from MCI,
"Our local calls have us in hock!" we all cry.

Oh Ma Bell why did you have to die?

 IMAGE_1        WINDOW_C       FRCH:019        Running       More..      F

The Ada design team was led by Jean D. Ichbiah and has included Bernd
Krieg-Brueckner, Brian A. Wichmann, Henry F. Ledgard, Jean-Claude Heliard,
Jean-Loup Gailly, Jean-raymond Abrial, John G.P. Barnes, Mike Woodger,
Olivier Roubine, Paul N. Hilfinger, and Robert Firth.

At various stages of the project, several people closely associated with{F4}
____________________________BOTTOM-OF-SCREEN__________________________________
```

Comments: The cursor has been positioned at the end of the viewport associated with WINDOW_C.

Command: {F4} - Display the next window full of text available for WINDOW_C.

SWITCH TO AIM WINDOW

```
 IMAGE_1         WINDOW_A        FRCH:017        Running                  F
    He chortled in his joy.

'Twas brillig, and the slithy toves
 Did gyre and gimble in the wabe.
All mimsy were the borogoves,
    And the mome raths outgrabe.

 IMAGE_1         WINDOW_B        FRCH:018        Running                  F

Bye, bye, Ma Bell, why did you die?
We get static from Sprint and echo from MCI,
"Our local calls have us in hock!" we all cry.

On Ma Bell why did you have to die?

 IMAGE_1         WINDOW_C        FRCH:019         Running                 F
Over the five years spent on this project, several intense week-long design
reviews were conducted, with the participation of P. Belmont, B. Brosgol,
P. Cohen, R. Dewar, A. Evans, G. Fisher, H. Harte, A.L. Hisgen, P. Knueven,
M. Kronental, N. Lomuto, E. Ploedereder, G. Seegmueller, V. Stenning, D.
Taffs, and also F. Belz, R. Converse, K. Correll, A.N. Habermann, J.
Sammet, S. Squires, J. Teller, P. Wegner, and P.R. Wetherall.{F10}
_____________________________BOTTOM-OF-SCREEN_____________________________
```

Comments: The listing of the "adalrm" file has finished, and the APSE command interpreter prompt is the last line of the viewport. Note the next page (window full) of output is displayed (continuously scrolled) in the viewport associated with WINDOW_C.

Command: {F10} - Switch the input context to the AIM window.

SET AIM_SUSPENDS

```
 AIM              AIM       AIM CLI        Running                        FIO
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> GOTO IMAGE IMAGE_1
AIM> RESET AIM_SUSPENDS
AIM> RESET FULL WINDOW_B
AIM> SET PADS AIM
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:AIM1.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:AIM2.OUT
AIM> SET INPUT_PAD WINDOW_A
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A3.INP
AIM> SET OUTPUT_PAD WINDOW_B
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A4.OUT
AIM> SET PADS WINDOW_C
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C5.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C6.OUT
AIM> GOTO IMAGE IMAGE_1
AIM> RESET INPUT_PAD WINDOW_A
AIM> RESET OUTPUT_PAD WINDOW_B
AIM> RESET PADS WINDOW_C
AIM> SET FULL WINDOW_B
AIM> GOTO WINDOW WINDOW_A
AIM> SET AIM_SUSPENDS
______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: Control returns to the AIM command interpreter in the AIM window.

Command: SET AIM_SUSPENDS - Set the SUSPEND_ON_AIM flag (i.e.--set up the condition that if and when the SWITCH_TO_AIM keystroke is entered, ALL APSE program output is suspended).

GOTO WINDOW_B

```
 AIM              AIM       AIM CLI        Running                         AFIO
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> GOTO IMAGE IMAGE_1
AIM> RESET AIM_SUSPENDS
AIM> RESET FULL WINDOW_B
AIM> SET PADS AIM
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:AIM1.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:AIM2.OUT
AIM> SET INPUT_PAD WINDOW_A
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A3.INP
AIM> SET OUTPUT_PAD WINDOW_B
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A4.OUT
AIM> SET PADS WINDOW_C
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C5.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C6.OUT
AIM> GOTO IMAGE IMAGE_1
AIM> RESET INPUT_PAD WINDOW_A
AIM> RESET OUTPUT_PAD WINDOW_B
AIM> RESET PADS WINDOW_C
AIM> SET FULL WINDOW_B
AIM> GOTO WINDOW WINDOW_A
AIM> SET AIM_SUSPENDS
AIM> GOTO WINDOW WINDOW_B
_______________________________BOTTOM-OF-SCREEN_________________________
```

Comments: The AIM window flag setting has been changed to reflect the fact that the SUSPEND_ON_AIM mode is active.

Command: GOTO WINDOW WINDOW_B - Switch the input context to WINDOW_B. Note since this is a global flag, it will always be ON or OFF for every window in the AIM environment.

LIST A FILE IN WINDOW_B

```
 IMAGE_1          WINDOW_A        FRCH:017        Running                      AF
    He chortled i his joy.

'Twas brillig, and the slithy toves
 Did gyre and gimble in the wabe:
All mimsy were the borogoves,
    And the mome raths outgrabe.

 IMAGE_1          WINDOW_B        FRCH:018        Running                      AF

Bye, bye, Ma Bell, why did you die?
We get static from Sprint and echo from MCI,
"Our local calls have us in hock!" we all cry.

Oh Ma Bell why did you have to die?
TYPE MABELL.AIM
 IMAGE_1          WINDOW_C        FRCH:019         Running                     AF
Over the five years spent on this project, several intense week-long design
reviews were conducted, with the participation of P. Belmont, B. Brosgol,
P. Cohen, R. Dewar, A. Evans, G. Fisher, H, Harte, A.L. Hisgen, P. Knueven,
M. Kronental, N. Lomuto, E. Ploedereder, G. Seegmueller, V. Stenning, D.
Taffs, and also F. Belz, R. Converse, K. Correll, A.N. Habermann, J.
Sammet, S. Squires, J. Teller, P. Wegner, and P.R. Wetherall.
____________________________BOTTOM-OF-SCREEN____________________________
```

Comments: The cursor is positioned at the end of the viewport associated with WINDOW_B.

Command: TYPE MABELL.AIM - List the contents of "mabell" in WINDOW_B.

SWITCH TO AIM WINDOW

```
________________________________________________________________________________
 IMAGE_1         WINDOW_A        FRCH:017        Running                       AF
    He chortled in his joy.

'Twas brillig, and the slithy toves
 Did gyre and gimble in the wabe:
All mimsy were the borogoves,
    And the mome raths outgrabe.

 IMAGE_1         WINDOW_B        FRCH:018        Running                       AF
"Our local calls have us in hock!" we all cry.

Oh Ma Bell why did you have to die?
TYPE MABELL.AIM
TYPE MABELL.AIM

         "The day Bell System Died"  by Lauren Weinstein {F10}
 IMAGE_1         WINDOW_C        FRCH:019         Running                      AF
Over the five years spent on this project, several intense week-long design
reviews were conducted, with the participation of P. Belmont, B. Brosgol,
P. Cohen, R. Dewar, A. Evans, G. Fisher, H. Harte, A.L. Hisgen, P. Knueven,
M. Kronental, N. Lomuto, E. Ploedereder, G. Seegmueller, V. Stenning, D.
Taffs, and also F. Belz, R. Converse, K. Correll, A.N. Habermann, J.
Sammet, S. Squires, J. Teller, P. Wegner, and P.R. Wetherall.
_____________________________BOTTOM-OF-SCREEN___________________________________
```

Comments: The first few lines of "mabell" are listed before the user enters the SWITCH_TO_AIM keystroke which suspends all APSE program output.

Command: {F10} - Switch the input context to the AIM window in order to communicate with the AIM command interpreter.

GOTO WINDOW_B

```
 AIM               AIM       AIM CLI       Running                          AFIO
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> RESET AIM_SUSPENDS
AIM> GOTO IMAGE IMAGE_1
AIM> RESET FULL WINDOW_B
AIM> SET PADS AIM
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:AIM1.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:AIM2.OUT
AIM> SET INPUT_PAD WINDOW_A
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A3.INP
AIM> SET OUTPUT_PAD WINDOW_B
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A4.OUT
AIM> SET PADS WINDOW_C
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C5.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C6.OUT
AIM> GOTO IMAGE IMAGE_1
AIM> RESET INPUT_PAD WINDOW_A
AIM> RESET OUTPUT_PAD WINDOW_B
AIM> RESET PADS WINDOW_C
AIM> SET FULL WINDOW_B
AIM> GOTO WINDOW WINDOW_A
AIM> RESET AIM_SUSPENDS
AIM> GOTO WINDOW WINDOW_B
AIM> GOTO WINDOW WINDOW_B
______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: Note the AIM window flag settings do not change as a result of the SWITCH_TO_AIM keystroke because no APSE program is ever associated with the AIM window.

Command: GOTO WINDOW WINDOW_B - Switch the input context to WINDOW_B.

SWITCH TO AIM WINDOW

```
_____________________________________________________________________________
 IMAGE_1        WINDOW_A       FRCH:017       Suspended                  AFS
    He chortled i his joy.

'Twas brillig, and the slithy toves
 Did gyre and gimble in the wabe:
All mimsy were the borogoves,
    And the mome raths outgrabe.

 IMAGE_1        WINDOW_B       FRCH:018       Suspended    More..        AFS

Long, long, time ago,
I can still remember,
Whem the local calls were "free".
And I knew if I paid my bill,
And never wished them any ill,
That the phone company would let me be...{F10}
 IMAGE_1        WINDOW_C       FRCH:019        Suspended                 AFS
Over the five years spent on this project, several intense week-long design
reviews were conducted, with the participation of P. Belmont, B. Brosgol,
P. Cohen, R. Dewar, A. Evans, G. Fisher, H. Harte, A.L. Hisgen, P. Knueven,
M. Kronental, N. Lomuto, E. Ploedereder, G. Seegmueller, V. Stenning, D.
Taffs, and also F. Belz, R. Converse, K. Correll, A.N. Habermann, J.
Sammet, S. Squires, J. Teller, P. Wegner, and P.R. Wetherall.
____________________________BOTTOM-OF-SCREEN_________________  ___________
```

Comments: The cursor is positioned at the end of the viewport associated with WINDOW_B. Note the viewport headers reflect the new window flag settings ("AFS"), as all APSE program output has been suspended. However, all information that was queued before the suspend flag was set will be output, thus the more flag is set and the queued data may still be output. To resume the APSE program output the user must use the RESUME PROGRAM_OUTPUT command (see the next session for details of this command).

Command: {F10} - Switch the input context to the AIM window in order to communicate with the AIM command interpreter.

ABORT THE AIM

```
 AIM               AIM      AIM CLI       Running                        AFIO
AIM> RESET AIM_SUSPENDS
AIM> GOTO IMAGE IMAGE_1
AIM> RESET FULL WINDOW_B
AIM> SET PADS AIM
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:AIM1.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:AIM2.OUT
AIM> SET INPUT_PAD WINDOW_A
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A3.INP
AIM> SET OUTPUT_PAD WINDOW_B
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A4.OUT
AIM> SET PADS WINDOW_C
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C5.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C6.OUT
AIM> GOTO IMAGE IMAGE_1
AIM> RESET INPUT_PAD WINDOW_A
AIM> RESET OUTPUT_PAD WINDOW_B
AIM> RESET PADS WINDOW_C
AIM> SET FULL WINDOW_B
AIM> GOTO WINDOW WINDOW_A
AIM> SET AIM_SUSPENDS
AIM> GOTO WINDOW WINDOW_B
AIM> GOTO WINDOW WINDOW_B
AIM> ABORT_AIM
______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: Control has returned to the AIM command interpreter in the AIM window.

Command: ABORT_AIM - Abort this AIM session (i.e.--kill the APSE LIST process running in WINDOW_B and then exit the AIM).

ABORT MESSAGE

```
AIM> RESET AIM_SUSPENDS
AIM> RESET FULL WINDOW_B
AIM> SET PADS AIM
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:AIM1.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:AIM2.OUT
AIM> SET INPUT_PAD WINDOW_A
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A3.INP
AIM> SET OUTPUT_PAD WINDOW_B
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A4.OUT
AIM> SET PADS WINDOW_C
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C5.INP
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C6.OUT
AIM> GOTO IMAGE IMAGE_1
AIM> RESET INPUT_PAD WINDOW_A
AIM> RESET OUTPUT_PAD WINDOW_B
AIM> RESET PADS WINDOW_C
AIM> SET FULL WINDOW_B
AIM> GOTO WINDOW WINDOW_A
AIM> SET AIM_SUSPENDS
AIM> GOTO WINDOW WINDOW_B
AIM> GOTO WINDOW WINDOW_B
AIM> ABORT_AIM
-)
________________________________BOTTOM-OF-SCREEN________________________________
```

Comments: The AIM command interpreter indicates that the APSE process executing in WINDOW_B has been stopped, and then the AIM terminates its execution.

Command: None.

The following pad (text) files were generated in this session:

1. :USER1:ATB:AIM:TESTING:AIM1.INP

```
set input_pad window_a
set output_pad window_b
set pads window_c
goto image image_1
reset input_pad window_a
reset output_pad window_b
reset pads window_c
set full window_b
goto window window_a
set aim_suspends
goto window window_b
goto window window_b
abort_aim
```

2. :USER1:ATB:AIM:TESTING:AIM2.OUT

```
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:AIM2.OUT
AIM>
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_A3.
AIM>
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_B4
AIM>
**AIM generated input pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C5.
**AIM generated output pad file name: :USER1:ATB:AIM:TESTING:WINDOW_C6
AIM>
AIM>
AIM>
AIM>
AIM>
AIM>
AIM>
AIM>
AIM>
AIM>
```

3. WINDOW_A3.INP

```
type jab.aim
```

4. WINDOW_B4.OUT

```
) type mabell.aim
     "The Day Bell System Died"  by Lauren Weinstein

Long, long, time ago,
I can still remember,
When the local calls were "free".
And I knew if I paid my bill,
And never wished them any ill,
That the phone company would let me be...
```

But Uncle Sam said he knew better,
Split 'em up, for all and ever!
We'll foster competition:
It's good capital-ism!

I can't remember if I cried,
When my phone bill first tripled in size.
But something touched me deep inside,
The day... Bell System... died.

And we were singing...

Bye, bye, Ma Bell, why did you die?
We get static from Sprint and echo from MCI,
"Our local calls have us in hock!" we all cry.
Oh Ma Bell why did you have to die?
Ma Bell why did you have to die?

Is your office Step by Step,
Or have you gotten some Crossbar yet?
Everybody used to ask...
Oh, is TSPS coming soon?
IDDD will be a boon!
And, I hope to get a Touch-Tone phone, real soon...

The color phones are really neat,
And direct dialing can't be beat!
My area code is "low":
The prestige way to go!

Oh, they just raised phone booths to a dime!
Well, I suppose it's about time.
I remember how the payphones chimed,
The day... Bell System... died.

And we were singing...

Bye, bye, Ma Bell, why did you die?
We get static from Sprint and echo from MCI,
"Our local calls have us in hock!" we all cry.
Oh Ma Bell why did you have to die?
Ma Bell why did you have to die?

Back then we were all at one rate,
Phone installs didn't cause debate,
About who'd put which wire where...
Installers came right out to you,
No "phone stores" with their ballyhoo,
And 411 was free, seemed very fair!

But FCC wanted it seems,
To let others skim long-distance creams,
No matter 'bout the locals,
They're mostly all just yokels!

And so one day it came to pass,

That the great Bell System did collapse,
In rubble now, we all do mass,
The day... Bell System... died.

So bye, bye, Ma Bell, why did you die?
We get static from Sprint and echo from MCI,
"Our local calls have us in hock!" we all cry.
Oh Ma Bell why did you have to die?
Ma Bell why did you have to die?

I drove on out to Murray Hill,
To see Bell Labs, some time to kill,
But the sign there said the Labs were gone.
I went back to my old CO,
Where I'd had my phone lines, years ago,
But it was empty, dark, and ever so forlorn...

No relays pulsed,
No data crooned,
No MF tones did play their tunes,
There wasn't a word spoken,
All carrier paths were broken...

And so that's how it all occurred,
Microwave horns just nests for birds,
Everything became so absurd,
The day... Bell System... died.

So bye, bye, Ma Bell, why did you die?
We get static from Sprint and echo from MCI,
"Our local calls have us in hock!" we all cry.
Oh Ma Bell why did you have to die?
Ma Bell why did you have to die?

We were singing:

Bye, bye, Ma Bell, why did you die?
We get static from Sprint and echo from MCI,
"Our local calls have us in hock!" we all cry.

Oh Ma Bell why did you have to die?

5. WINDOW_C5.INP
   type lrm.aim

6. WINDOW_C6.OUT

```
                              Foreword


Ada is the result of a collective effort to design a  common  language  fc
programming large scale and real-time systems.

The  common high order language program began in 1974.  The requirements o
the United States Department of Defense were  formalized  in  a  series  o
documents  which  were  extensively  reviewed  by  the Services, industria
organizations, universities, and foreign  military  departments.   The  Ad
language  was  designed  in  accordance with the final (1978) form of thes
requirements, embodied in the Steelman specification.

The Ada design team was led  by Jean D.  Ichbiah  and  has  included  Bern
Krieg-Brueckner,  Brian A. Wichmann, Henry F. Ledgard, Jean-Claude Heliard,
Jean-Loup Gailly, Jean-Raymond Abrial,  John  G.P.  Barnes,  Mike  Woodger,
Olivier Roubine, Paul N. Hilfinger, and Robert Firth.

At various stages of the project, several people closely associated with
```

A.5 SESSION 5 - APSE PROGRAM RELATED COMMANDS

This session demonstrates the use of the SUSPEND, RESUME, and TERMINATE commands. The following text files are used in this session:

1. demo1.aim

```
CREATE  IMAGE   IMAGE_1
CREATE  IMAGE   IMAGE_2
CREATE  WINDOW  WINDOW_A
CREATE  WINDOW  WINDOW_B
CREATE  WINDOW  WINDOW_C
CREATE  WINDOW  WINDOW_D
ASSOC  WINDOW_A  IMAGE_1  1    8
ASSOC  WINDOW_B  IMAGE_1  9    8
ASSOC  WINDOW_C  IMAGE_1  17   8
ASSOC  WINDOW_C  IMAGE_2  1    12
ASSOC  WINDOW_D  IMAGE_2  13   12
```

2. demo15.aim

```
SCRIPT demo1.aim
SCRIPT demo16.aim
```

3. demo16.aim

```
GOTO  WINDOW  WINDOW_A
SUSPEND  EXECUTION  WINDOW_A
SUSPEND  PROGRAM_OUTPUT  WINDOW_C
GOTO  IMAGE  IMAGE_1
INFO  IMAGE  IMAGE_1
RESUME  EXECUTION  WINDOW_A
RESUME  PROGRAM_OUTPUT  WINDOW_C
GOTO  IMAGE  IMAGE_1
ABORT
```

4. your_file.aim

This file contains information concerning the eventual destruction and subsequent desolation of the world. The information contained herein is not of a secret matter but possibly could alarm the general population. I ask only that those who are exposed to this material use good judgement and be very cautious when discussing the contents with other people. For those who may be offended by all or part of the information contained herein, I offer my sincere apologies. This paper started as my personal thoughts on

INVOKE AIM FROM APSE

```
-) AIM











_____________________________BOTTOM-OF-SCREEN_____________________________
```

Comments:

Command: AIM - Invoke the AIM from the APSE.

SWITCH TO AIM IMAGE

```
____________________________________________________________________________
 MAIN           MAIN          FRCH:016       Running                    AF

AOS/VS CLI REV 05.01.00.00        9-APR-85     13:26:43 {F10}







__________________________BOTTOM-OF-SCREEN__________________________________
```

Comments: The AIM initializes the current image to be MAIN. To enter AIM textual commands, the user must communicate through the AIM window to the AIM command interpreter.

Command: {F10} - Set the current image to be the AIM image.

INVOKE SCRIPT

```
_________________________________________________________________________
 AIM           AIM      AIM CLI      Running                          AF
AIM> SCRIPT "demo15.aim"










______________________________BOTTOM-OF-SCREEN__________________________
```

Comments: The cursor is now at the top of the AIM image. AIM commands ca
only be entered in the AIM image. Control has been passed to the AIM C
executing in the AIM window.

Command: SCRIPT "demo15.aim" - Demo15 is a system dependent name.

ECHO SCRIPT INPUT

```
 AIM            AIM      AIM CLI       Running                         AF
AIM> SCRIPT "demo15.aim"
AIM> SCRIPT "demo1.aim"
AIM> CREATE  IMAGE   IMAGE_1
AIM> CREATE  IMAGE   IMAGE_2
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> CREATE  WINDOW  WINDOW_D
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> ASSOC  WINDOW_C  IMAGE_2  1   12
AIM> ASSOC  WINDOW_D  IMAGE_2  13  12
AIM> SCRIPT "demo16.aim"
AIM> GOTO  WINDOW  WINDOW_A




                              BOTTOM-OF-SCREEN
```

Comments: This session starts with the invocation of a script, demo15. Script demo15 invokes demo1. Demo1 creates some windows and images and associates them. Demo15 then invokes another script, demo16, to demonstrate the commands explained in this session.

Command: GOTO WINDOW WINDOW_A - This command is part of the script. The GOTO stops the script and puts the user in interactive mode in WINDOW_A.

INVOKE A PROGRAM

```
 IMAGE_1        WINDOW_A       FRCH:017       Running                    AF

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29 ENTER/NO_NEWS ^^



 IMAGE_1        WINDOW_B       FRCH:018       Running                    AF

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29



 IMAGE_1        WINDOW_C       FRCH:019        Running                    AF

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29



____________________________BOTTOM-OF-SCREEN________________________________
```

Comments: Demol completes execution and demol6 starts. Demol6 contains a GOTO which sends the cursor to one of the windows created in demol.

Command: ENTER/NO_NEWS - enter the ADE to compile a program

INVOKE A PROGRAM

```
 IMAGE_1         WINDOW_A        FRCH:017        Running                     AF
AOS/VS CLI   REV 05.01.00.00        11-APR-85    8:55:29 ENTER/NO_NEWS ^^
)  ENTER/NO_NEWS ^^

    DATA GENERAL/ROLM  Ada  Development  Environment  Revision  2.20.00.00

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
 IMAGE_1         WINDOW_B        FRCH:018        Running                     AF

AOS/VS CLI   REV 05.01.00.00        11-APR-85    8:55:29




 IMAGE_1         WINDOW_C        FRCH:019         Running                     AF

AOS/VS CLI   REV 05.01.00.00        11-APR-85    8:55:29



______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The ENTER command is used to enter the ADE. The ADE was entered two directory levels above the source code to be compiled.

Command: DIR REHOST:TESTING - reset the directory

INVOKE A PROGRAM

```
 IMAGE_1         WINDOW_A        FRCH:017        Running                      AF
)  ENTER/NO_NEWS ^^

    DATA GENERAL/ROLM  Ada  Development  Environment  Revision  2.20.00.00

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
-  DIR REHOST:TESTING BATCH ADA/MAIN_PROGRAM TEST
 IMAGE_1         WINDOW_B        FRCH:018        Running                      AF

AOS/VS CLI   REV 05.01.00.00        11-APR-85    8:55:29




 IMAGE_1         WINDOW_C        FRCH:019        Running                      AF

AOS/VS CLI   REV 05.01.00.00        11-APR-85    8:55:29




____________________________BOTTOM-OF-SCREEN____________________________
```

Comments: The directory command is used to return to the source code to be compiled.

Command: BATCH ADA/MAIN_PROGRAM TEST - Execute the Ada compiler.

INVOKE A PROGRAM

```
 IMAGE_1          WINDOW_A          FRCH:017          Running                    AF

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
-  DIR REHOST:TESTING BATCH ADA/MAIN_PROGRAM TEST
-  BATCH ADA/MAIN_PROGRAM TEST
DELETED :udd:FRCH:batch:batch_out
QUEUED, SEQ 18533, QPRI=127 {F5}
 IMAGE_1          WINDOW_B          FRCH:018          Running                    AF

AOS/VS CLI    REV 05.01.00.00        11-APR-85    8:55:29



 IMAGE_1          WINDOW_C          FRCH:019          Running                    AF

AOS/VS CLI    REV 05.01.00.00        11-APR-85    8:55:29



______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The Ada compiler is started.

Command: {F5}

EXECUTE ANOTHER PROGRAM

```
 IMAGE_1         WINDOW_A        FRCH:017        Running                    AF

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
-  DIR REHOST:TESTING BATCH ADA/MAIN_PROGRAM TEST
-  BATCH ADA/MAIN_PROGRAM TEST
DELETED :udd:FRCH:batch:batch_out
QUEUED, SEQ 18533, QPRI=127
 IMAGE_1         WINDOW_B        FRCH:018        Running                    AF

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29 ENTER/NO_NEWS ^^




 IMAGE_1         WINDOW_C        FRCH:019        Running                    AF

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29




_____________________________BOTTOM-OF-SCREEN_________________________________
```

Comments: The cursor has now moved to the next viewport using the default binding for the NEXT_VIEWPORT keystroke.

Command: ENTER/NO_NEWS - Enter the ADE two levels up again program.

EXECUTE ANOTHER PROGRAM

```
 IMAGE_1         WINDOW_A        FRCH:017        Running                    AF

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
-  DIR REHOST:TESTING BATCH ADA/MAIN_PROGRAM TEST
-  BATCH ADA/MAIN_PROGRAM TEST
DELETED :udd:FRCH:batch:batch_out
QUEUED, SEQ 18533, QPRI=127
 IMAGE_1         WINDOW_B        FRCH:018         Running                    AF
AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29 ENTER/NO_NEWS ^^
)  ENTER/NO_NEWS ^^

    DATA GENERAL/ROLM  Ada  Development  Environment  Revision  2.20.00.00

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
 IMAGE_1         WINDOW_C        FRCH:019         Running                    AF

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29




____________________________BOTTOM-OF-SCREEN____________________________
```

Comments: The ENTER command must be issued for each window.

Command: DIR REHOST:TESTING - again, reset the directory program.

EXECUTE ANOTHER PROGRAM

```
 IMAGE_1         WINDOW_A        FRCH:017        Running                       AF

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
-  DIR REHOST:TESTING BATCH ADA/MAIN_PROGRAM TEST
-  BATCH ADA/MAIN_PROGRAM TEST
DELETED :udd:FRCH:batch:batch_out
QUEUED, SEQ 18533, QPRI=127
 IMAGE_1         WINDOW_B        FRCH:018         Running                       AF
)  ENTER/NO_NEWS ^^

    DATA GENERAL/ROLM  Ada  Development  Environment  Revision  2.20.00.00

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
-  DIR REHOST:TESTING BATCH ADALINK/MAIN_PROGRAM TEST
 IMAGE_1         WINDOW_C        FRCH:019         Running                       AF

AOS/VS CLI    REV 05.01.00.00       11-APR-85    8:55:29



_____________________________BOTTOM-OF-SCREEN_____________________________
```

Comments: Return to the directory of the compiled source to link it.

Command: BATCH ADALINK/MAIN_PROGRAM TEST - Execute the Linker program.

EXECUTE ANOTHER PROGRAM

```
 IMAGE_1           WINDOW_A          FRCH:017          Running                         AF

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
-  DIR REHOST:TESTING BATCH ADA/MAIN_PROGRAM TEST
-  BATCH ADA/MAIN_PROGRAM TEST
DELETED :udd:FRCH:batch:batch_out
QUEUED, SEQ 18533, QPRI=127
 IMAGE_1           WINDOW_B          FRCH:018           Running                        AF

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
-  DIR REHOST:TESTING BATCH ADALINK/MAIN_PROGRAM TEST
-  BATCH ADALINK/MAIN_PROGRAM TEST
DELETED :udd:FRCH:batch:batch_out
QUEUED, SEQ=18537, QPRI=127 {F5}
 IMAGE_1           WINDOW_C          FRCH:019           Running                        AF

AOS/VS CLI    REV 05.01.00.00       11-APR-85     8:55:29




_____________________________BOTTOM-OF-SCREEN_____________________________
```

Comments: The cursor has now moved to the next viewport using the default binding for the NEXT_VIEWPORT keystroke.

Command: BATCH ADALINK/MAIN_PROGRAM TEST - Execute the Linker program.

EXECUTE A THIRD PROGRAM

```
 IMAGE_1          WINDOW_A        FRCH:017       Running                      AF

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
-  DIR REHOST:TESTING BATCH ADA/MAIN_PROGRAM TEST
-  BATCH ADA/MAIN_PROGRAM TEST
DELETED :udd:FRCH:batch:batch_out
QUEUED, SEQ 18533, QPRI=127
 IMAGE_1          WINDOW_B        FRCH:018        Running                      AF

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
-  DIR REHOST:TESTING BATCH ADALINK/MAIN_PROGRAM TEST
-  BATCH ADALINK/MAIN_PROGRAM TEST
DELETED :udd:FRCH:batch:batch_out
QUEUED, SEQ=18537, QPRI=127
 IMAGE_1          WINDOW_C        FRCH:019        Running                      AF

AOS/VS CLI   REV 05.01.00.00       11-APR-85    8:55:29 TYPE YOUR_FILE.AIM




____________________________BOTTOM-OF-SCREEN____________________________
```

Comments: The cursor has moved to the last window on the image. The compiler and linker are both running. (It is assumed that the programs being compiled and linked require a sufficient amount of time to present these examples.)

Command: TYPE YOUR_FILE.AIM

SWITCH BACK TO AIM IMAGE

```
 IMAGE_1        WINDOW_A       FRCH:017       Running                 AF

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
-  DIR REHOST:TESTING BATCH ADA/MAIN_PROGRAM TEST
-  BATCH ADA/MAIN_PROGRAM TEST
DELETED :udd:FRCH:batch:batch_out
QUEUED, SEQ 18533, QPRI=127
 IMAGE_1        WINDOW_B       FRCH:018        Running                 AF

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
-  DIR REHOST:TESTING BATCH ADALINK/MAIN_PROGRAM TEST
-  BATCH ADALINK/MAIN_PROGRAM TEST
DELETED :udd:FRCH:batch:batch_out
QUEUED, SEQ=18537, QPRI=127
 IMAGE_1        WINDOW_C       FRCH:019        Running                 AF

AOS/VS CLI   REV 05.01.00.00      11-APR-85    8:55:29 TYPE YOUR_FILE.AIM
{F10}



_____________________________BOTTOM-OF-SCREEN____________________________
```

Comments: Now start a third program. The List program is executed to list a file which contains text for this demonstration.

Command: {F10} - SWITCH_TO_AIM default keystroke. Return to AIM image using the default binding for the SWITCH_TO_AIM keystroke.

SUSPENDING PROGRAM OUTPUT AND EXECUTION

```
 AIM               AIM       AIM CLI        Running                        AF
AIM> SCRIPT "demo15.aim"
AIM> SCRIPT "demo1.aim"
AIM> CREATE  IMAGE  IMAGE_1
AIM> CREATE  IMAGE  IMAGE_2
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> CREATE  WINDOW  WINDOW_D
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> ASSOC  WINDOW_C  IMAGE_2  1   12
AIM> ASSOC  WINDOW_D  IMAGE_2  13  12
AIM> SCRIPT "demo16.aim"
AIM> GOTO  WINDOW  WINDOW_A
AIM> SUSPEND  EXECUTION  WINDOW_A
AIM> SUSPEND  PROGRAM_OUTPUT  WINDOW_C
AIM> GOTO  IMAGE  IMAGE_1




______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: Three programs have been started; one for each window in IMAGE_1. When switched back to the AIM image, the script resumes.

Command: GOTO IMAGE IMAGE_1 - This command is a part of the script. The GOTO puts the cursor back IMAGE_1.

CHECK RESULTS OF SUSPEND COMMANDS

```
 IMAGE_1          WINDOW_A        FRCH:017        Suspended                     AFS

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
-  DIR REHOST:TESTING BATCH ADA/MAIN_PROGRAM TEST
-  BATCH ADA/MAIN_PROGRAM TEST
DELETED :udd:FRCH:batch:batch_out
QUEUED, SEQ 18533, QPRI=127
 IMAGE_1          WINDOW_B        FRCH:018        Suspended                     AFS

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
-  DIR REHOST:TESTING BATCH ADALINK/MAIN_PROGRAM TEST
-  BATCH ADALINK/MAIN_PROGRAM TEST
DELETED :udd:FRCH:batch:batch_out
QUEUED, SEQ=18537, QPRI=127
 IMAGE_1          WINDOW_C        FRCH:019         Suspended                     AFS
subsequent desolation of the world.  The information contained herein is not
of a secret matter but possibly could alarm the general population.  I ask
only that those who are exposed to this material use good judgement and be
very cautious when discussing the contents with other people.  For those
who may be offended by all or part of the information contained herein, I
offer my sincere apologies.  This paper started as my personal thoughts on
{F10}
_____________________________BOTTOM-OF-SCREEN_______________________________
```

Comments: The script contains commands to suspend execution and window output and a GOTO. The GOTO puts the cursor back to IMAGE_1 so the results of the suspend commands can be seen. Note that the program in WINDOW_A has been suspended. The user can see that the header contains "Suspended" instead of "Running" indicating that window output has been suspended.

Command: {F10} - Switch back to AIM image. Enter the SWITCH_TO_AIM keystroke again.

MORE INFO

```
 AIM            AIM    AIM CLI        Running      More..          AF
AIM> SCRIPT "demo1.aim"
AIM> CREATE  IMAGE  IMAGE_1
AIM> CREATE  IMAGE  IMAGE_2
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> CREATE  WINDOW  WINDOW_D
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> ASSOC  WINDOW_C  IMAGE_2  1   12
AIM> ASSOC  WINDOW_D  IMAGE_2  13  12
AIM> SCRIPT "demo16.aim"
AIM> GOTO  WINDOW  WINDOW_A
AIM> SUSPEND  EXECUTION  WINDOW_A
AIM> SUSPEND  PROGRAM_OUTPUT  WINDOW_C
AIM> GOTO  IMAGE  IMAGE_1
AIM> INFO  IMAGE  IMAGE_1

Image Name: IMAGE_1

Window Associations:
Window Name         Top Line      Viewport Length{F4}
______________________________BOTTOM-OF-SCREEN_________________________
```

Comments: More flag set

Command: {F4} - next page keystroke

EXIT FIRST LEVEL OF INFO

```
 AIM                 AIM         AIM CLI        Running                       AF
AIM> CREATE  WINDOW  WINDOW_C
AIM> CREATE  WINDOW  WINDOW_D
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> ASSOC  WINDOW_C  IMAGE_2  1   12
AIM> ASSOC  WINDOW_D  IMAGE_2  13  12
AIM> SCRIPT "demo16.aim"
AIM> GOTO  WINDOW  WINDOW_A
AIM> SUSPEND  EXECUTION  WINDOW_A
AIM> SUSPEND  PROGRAM_OUTPUT  WINDOW_C
AIM> GOTO  IMAGE  IMAGE_1
AIM> INFO  IMAGE  IMAGE_1

Image Name: IMAGE_1

Window Associations:
Window Name         Top Line       Viewport Length
WINDOW_A               1                  8
WINDOW_B               9                  8
WINDOW_C               17                 8

INFO Image Name? {return}
_______________________________BOTTOM-OF-SCREEN_____________________________
```

Comments: The script contains an INFO command to allow another way of checking the status of the programs running in the windows of IMAGE_1.

Command: {return} - Enter a carriage return to exit the current level of the INFO utility.

EXIT INFO

```
 AIM              AIM       AIM CLI          Running                          AF
AIM> CREATE  WINDOW  WINDOW_D
AIM> ASSOC  WINDOW_A  IMAGE_1  1    8
AIM> ASSOC  WINDOW_B  IMAGE_1  9    8
AIM> ASSOC  WINDOW_C  IMAGE_1  17   8
AIM> ASSOC  WINDOW_C  IMAGE_2  1    12
AIM> ASSOC  WINDOW_D  IMAGE_2  13   12
AIM> SCRIPT "demo16.aim"
AIM> GOTO  WINDOW  WINDOW_A
AIM> SUSPEND  EXECUTION  WINDOW_A
AIM> SUSPEND  PROGRAM_OUTPUT  WINDOW_C
AIM> GOTO  IMAGE  IMAGE_1
AIM> INFO  IMAGE  IMAGE_1

Image Name: IMAGE_1

Window Associations:
Window Name          Top Line      Viewport Length
WINDOW_A                1                 8
WINDOW_B                9                 8
WINDOW_C                17                8

INFO Image Name?
INFO Object? {return}
______________________________BOTTOM-OF-SCREEN____________________________
```

Comments: Entering a carriage return exits the current level of INFO command.

Command: {return} - Enter a second carriage return to exit INFO.

SCRIPT RESUMES AFTER EXITING INFO

```
 AIM               AIM       AIM CLI       Running                      AF
AIM> ASSOC  WINDOW_C  IMAGE_2  1   12
AIM> ASSOC  WINDOW_D  IMAGE_2  13  12
AIM> SCRIPT "demo16.aim"
AIM> GOTO  WINDOW  WINDOW_A
AIM> SUSPEND  EXECUTION  WINDOW_A
AIM> SUSPEND  PROGRAM_OUTPUT  WINDOW_C
AIM> GOTO  IMAGE  IMAGE_1
AIM> INFO  IMAGE  IMAGE_1

Image Name: IMAGE_1

Window Associations:
Window Name          Top Line      Viewport Length
WINDOW_A               1                 8
WINDOW_B               9                 8
WINDOW_C               17                8

INFO Image Name?
INFO Object?

AIM> RESUME  EXECUTION  WINDOW_A
AIM> RESUME  PROGRAM_OUTPUT  WINDOW_C
AIM> GOTO  IMAGE  IMAGE_1
______________________________BOTTOM-OF-SCREEN____________________________
```

Comments: Exit the last level of INFO to restart the script.

Command: GOTO IMAGE IMAGE_1 - This command is contained in the script. The GOTO puts the cursor once again to IMAGE_1.

CHECK STATUS OF WINDOWS

```
 IMAGE_1          WINDOW_A        FRCH:017        Running                    AF

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
-  DIR REHOST:TESTING BATCH ADA/MAIN_PROGRAM TEST
-  BATCH ADA/MAIN_PROGRAM TEST
DELETED :udd:FRCH:batch:batch_out
QUEUED, SEQ 18533, QPRI=127
 IMAGE_1          WINDOW_B        FRCH:018        Running                    AF

ADE Directory is :USER1:ADE
Current directory is :USER1:ATB:AIM DIR REHOST:TESTING
-  DIR REHOST:TESTING BATCH ADALINK/MAIN_PROGRAM TEST
-  BATCH ADALINK/MAIN_PROGRAM TEST
DELETED :udd:FRCH:batch:batch_out
QUEUED, SEQ=18537, QPRI=127
 IMAGE_1          WINDOW_C        FRCH:019         Running                   AF
subsequent desolation of the world.  The information contained herein is no
of a secret matter but possibly could alarm the general population.  I ask
only that those who are exposed to this material use good judgement and be
very cautious when discussing the contents with other people.  For those
who may be offended by all or part of the information contained herein, I
offer my sincere apologies.  This paper started as my personal thoughts on
{F10}
_____________________________BOTTOM-OF-SCREEN_________________________________
```

Comments: The script resumes execution when the info utility is exited. The script contains the RESUME commands and also a TERMINATE command. The last line of the script is a GOTO command. Go back to the image and check the results of these commands. Note that the program in WINDOW_A is no longer suspended and has now completed. (The TERMINATE command has not been fully implemented.)

Command: {F10} - Switch to AIM image and exit.

ABORT

```
 AIM             AIM      AIM CLI       Running                         AF
AIM> ASSOC  WINDOW_C  IMAGE_2  1   12
AIM> ASSOC  WINDOW_D IMAGE_2  13  12
AIM> SCRIPT "demo16.aim"
AIM> GOTO WINDOW WINDOW_A
AIM> SUSPEND  EXECUTION  WINDOW_A
AIM> SUSPEND  PROGRAM_OUTPUT  WINDOW_C
AIM> GOTO  IMAGE  IMAGE_1
AIM> INFO  IMAGE  IMAGE_1

Image Name: IMAGE_1

Window Associations:
Window Name         Top Line      Viewport Length
WINDOW_A               1                 8
WINDOW_B               9                 8
WINDOW_C               17                8

INFO Image Name?
INFO Object?
AIM> RESUME  EXECUTION  WINDOW_A
AIM> RESUME  PROGRAM_OUTPUT  WINDOW_C
AIM> GOTO  IMAGE  IMAGE_1
AIM> ABORT_AIM
______________________________BOTTOM-OF-SCREEN_____________________________
```

Comments: Go back to the AIM image to exit.

Command: ABORT_AIM - This command is contained in the script. Abort the AIM and all sub-processes.

ABORT

```
AIM> ASSOC  WINDOW_C  IMAGE_2  1   12
AIM> ASSOC WINDOW_D IMAGE_2  13  12
AIM> SCRIPT "demo16.aim"
AIM> GOTO WINDOW WINDOW_A
AIM> SUSPEND  EXECUTION  WINDOW_A
AIM> SUSPEND  PROGRAM_OUTPUT  WINDOW_C
AIM> GOTO  IMAGE  IMAGE_1
AIM> INFO  IMAGE  IMAGE_1

Image Name: IMAGE_1

Window Associations:
Window Name          Top Line       Viewport Length
WINDOW_A               1                   8
WINDOW_B               9                   8
WINDOW_C               17                  8

INFO Image Name?
INFO Object?
AIM> RESUME  EXECUTION  WINDOW_A
AIM> RESUME  PROGRAM_OUTPUT  WINDOW_C
AIM> GOTO  IMAGE  IMAGE_1
AIM> ABORT_AIM
-)
______________________________BOTTOM-OF-SCREEN____________________________
```

Comments: The ABORT command terminates all active programs. The progra
running in WINDOW_C is aborted.

A.6 SESSION 6 - AIM HELP UTILITY

The AIM Help Utility session explains the use of the HELP command. The Help Utility presents information on the AIM commands and their associated parameters. The following text files are used in this session:

None.

INVOKE AIM FROM APSE

```
-) AIM
```

```
_____________________________BOTTOM-OF-SCREEN_____________________________
```

Comments:

Command: AIM - invoke the AIM from the APSE.

SWITCH TO AIM IMAGE

```
 MAIN           MAIN           FRCH:016       Running                      AF

AOS/VS CLI REV 05.01.00.00        9-APR-85     13:26:43 {F10}




















__________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The AIM initializes the current image to be MAIN. To enter AIM textual commands the user must communicate through the AIM image to the AIM command interpreter.

Command: {F10} - Set the current image to be the AIM image.

INVOKE DEMO1.AIM SCRIPT

```
 AIM                AIM        AIM CLI         Running                         AF
AIM> HELP
















______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The AIM image is initially blank.

Command: HELP - invoke AIM HELP utility.

THE AIM HELP UTILITY

```
 AIM             AIM       AIM CLI        Running                       AF
AIM> HELP
Initializing Help...

Information Available:

ABORT_AIM                      INFO
ASSOCIATE                      KEYSTROKES
CREATE                         RESET
DEFINE                         RESUME
DELETE                         SCRIPT
DISASSOCIATE                   SET
EXIT                           SUSPEND
GOTO                           TERMINATE_EXECUTION
HELP

Topic? A




__________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The HELP utility lists ALL the AIM topics for which help is available. When HELP prompts for a topic name, key in one of the AIM keywords from the list.

Command: A - request HELP information on ALL AIM commands (textual or keystroke) that start with the single character 'A'.

MATCH THE 'A' ABBREVIATION

```
 AIM               AIM       AIM CLI         Running       More..              AF
Initializing Help...

Information Available:

ABORT_AIM                            INFO
ASSOCIATE                            KEYSTROKES
CREATE                               RESET
DEFINE                               RESUME
DELETE                               SCRIPT
DISASSOCIATE                         SET
EXIT                                 SUSPEND
GOTO                                 TERMINATE_EXECUTION
HELP

Topic? A

ABORT_AIM

      The ABORT_AIM command terminates all processes running under the AIM
      and exits from the AIM program.  All active subordinate processes are
      terminated without warning.

Additional Information Available: {F4}
_________________________________BOTTOM-OF-SCREEN___________________________
```

Comments: Three AIM commands were recognized from the 'A'. The corresponding top level HELP text for each command is printed alphabetically by topic name. The HELP text did NOT all fit in one screenfull, so the "More.." indicator is turned on in the viewport header.

Command: {F4} - display next page of window's output.

REMAINDER OF HELP TEXT FOR THE 'A' ABBREVIATION

```
______________________________________________________________________________
 AIM            AIM      AIM CLI      Running      More..            AF
SYNTAX                            EXAMPLES

ASSOCIATE

     The ASSOCIATE command maps a specified portion of a window onto an
     image.  Assuming that n = <length> and p = <position>, the ASSOCIATE
     command maps the last n lines of the specified window onto the given
     image starting at the pth line of the image.  For example,

                         ASSOC WIN_1 IM_1 13 12

     will map the last 12 lines of WIN_1 onto lines 13..24 of IM_1.  This
     forms an abstract association between WIN_1 and IM_1 by creating a
     viewport (lines 13 thru 24) onto which the last 12 lines of WIN_1 are
     mapped.  This viewport partitions the image vertically, and its width is
     equal to the width of the image.

     Note: A window may be associated with more than one image at the same
     time; however, a specific window may only be associated with a specific
     image once.

Additional Information Available:
{F4}
__________________________BOTTOM-OF-SCREEN_____________________________
```

Comments: More help

Command: {F4}

MORE HELP

```
 AIM              AIM        AIM CLI        Running                          AF
    The ASSOCIATE command maps a specified portion of a window onto an
    image.  Assuming that n = <length> and p = <position>, the ASSOCIATE
    command maps the last n lines of the specified window onto the given
    image starting at the pth line of the image.  For example,

                        ASSOC WIN_1 IM_1 13 12

    will map the last 12 lines of WIN_1 onto lines 13..24 of IM_1.  This
    forms an abstract association between WIN_1 and IM_1 by creating a
    viewport (lines 13 thru 24) onto which the last 12 lines of WIN_1 are
    mapped.  This viewport partitions the image vertically, and its width
    equal to the width of the image.

    Note: A window may be associated with more than one image at the same
    time; however, a specific window may only be associated with a specifi
    image once.

Additional Information Available:

SYNTAX                                      EXAMPLES
COMMAND_PARAMETERS                          ERRORS

topic? ASSOC *
___________________________BOTTOM-OF-SCREEN________________________________
```

Comments: Since a UNIQUE AIM topic was NOT matched the HELP utility prompts for another topic. In other words, the HELP utility could NOT begin a traversal of the HELP tree since a uniquely path was NOT identified.

Command: ASSOC * - "ASSOC" uniquely identifies the ASSOCIATE command. "*" queries for ALL available HELP information supported for the ASSOCIATE command. This includes the top level ASSOCIATE information along with information for all subtopics.

FETCH ALL INFO ON ASSOC

```
________________________________________________________________________________
 AIM               AIM        AIM CLI       Running        More..            AF
COMMAND_PARAMETERS                  ERRORS

topic? ASSOC *

ASSOCIATE

     The ASSOCIATE command maps a specified portion of a window onto an
     image. Assuming that n = <length> and p = <position>, the  ASSOCIATE
     command maps the last n lines of the specified window onto the given
     image starting at the pth line of the image.  For example,

                              ASSOC WIN_1 IM_1 13 12

     will map the last 12 lines of WIN_1 onto lines 13..24 of IM_1. This
     forms an abstract association between WIN_1 and IM_1 by creating a
     viewport (lines 13 thru 24) onto which the last 12 lines of WIN_1 are
     mapped. This viewport partitions the image vertically, and its width is
     equal to the width of the image.

     Note: A window may be associated with more than one image at the same
     time; however, a specific window may only be associated with a specific
     image once.
{F4}
________________________________BOTTOM-OF-SCREEN________________________________
```

Comments: The HELP information has filled up the AIM window. The "More.." indicator is on in the viewport header. This informs the user that the window output has been suspended because the window is full and the suspend on full window output mode is set ("F" setting in viewport header). The HELP utility prompts the user for a carriage return in order to display the next page of the window's output.

Command: {F10} - display next page of window's output.

```
                          FETCH ALL INFO ON ASSOCIATE
______________________________________________________________________
 AIM            AIM       AIM CLI       Running       More..         AF

SYNTAX                          EXAMPLES
COMMAND_PARAMETERS              ERRORS

SYNTAX

     AS*SOCIATE WINDOW_NAME => <window_name>
                IMAGE_NAME  => <image_name>
                TOP         => <position>
                LENGTH      => <length>

COMMAND_PARAMETERS

             WINDOW_NAME => <window_name>

     Specifies the window whose last <length> lines  are to be mapped onto
     the specified image.

               IMAGE_NAME => <image_name>

     Specifies the image onto which a portion of the given window will be
     mapped.
{F4}
___________________________BOTTOM-OF-SCREEN___________________________


Comments:  More on the ASSOCIATE topic

Command:  {F4}
```

```
                                MORE HELP
_____________________________________________________________________________
 AIM             AIM      AIM CLI       Running      More..           AF

      Specifies the starting position for the association relative to the to
      of the specified image.

              LENGTH => <length>

      Specifies the length of the viewport used for the requested
      association. In addition to the viewport header, at least one line of
      window must be displayed in a viewport; therefore, the minimum length
      allowable is 2.

EXAMPLES

      Long Form:       AIM> ASSOCIATE WINDOW_NAME => W_1
                                       IMAGE_NAME => I_1
                                              TOP => 1
                                           LENGTH => 24

      Associate the entire contents of window W_1 with the image I_1.

      Mixed Form:      AIM> ASSOC WIN_2 IMAGE_NAME => IM_2 1 LENGTH => 8

      Associate the last 8 lines of window WIN_2 with the first 8 lines of
___________________________BOTTOM-OF-SCREEN__________________________________


Comments:  More on ASSOCIATE

Command:  {F4}
```

```
                              More flag set
 _______________________________________________________________________
  AIM            AIM       AIM CLI        Running      More.            AF
        Short Form:      AIM> AS WIN_3 IM_3 9 8

        Associate the last 8 lines of window WIN_3 with lines 9 thru 16 of
        image IM_1.

 ERRORS

        The semantic errors associated with this command include:

                "Window <window_name> does not exist"

        The specified window name, <window_name>, does NOT exist (i.e.-- no
        window of that name has been created).

                "Image <image_name> does not exist"

        The specified image name, <image_name>, does NOT exist (i.e.-- no im
        of that name has been created).

                "Invalid length: <length>"

        The specified viewport length was wither out of range, non-numeric,
 _________________________BOTTOM-OF-SCREEN______________________________
```

Comments: MORE HELP

Command: {F4}

MORE HELP

```
AIM               AIM       AIM CLI       Running                            AF
        The specified image name, <image_name>, does NOT exist (i.e.--no image
        of that name has been created).

                  "Invalid length: <length>"

        The specified viewport length was either out of range, non-numeric, or
        too long to ensure non-intersecting viewports of the given image.

                  "Invalid top line: <top_line>"

        The specified top line for starting the viewport was either out of
        range, or non-numeric.

                  "Association between specified window and image already exists

        The specified window is currently associated with <image_name> and the
        AIM prohibits multiple associations between the same window and image.

                  "Cannot map another window onto the AIM image"

        The AIM image and window cannot be altered by the user.

ASSOCIATE subtopic? ?
_______________________________BOTTOM-OF-SCREEN___________________________
```

Comments: Upon the carriage return the remainder of the HELP text is displayed and the "More.." indicator is NO longer present. The "ASSOC *" HELP command matched ASSOCIATE, so the "ASSOCIATE Subtopic?" appears.

Command: ? - invoke the implicit help operator to display a list of allowable responses to the "ASSOCIATE Subtopic" prompt.

IMPLICIT HELP AT ASSOCIATE SUBTOPIC LEVEL

```
___________________________________________________________________________
AIM           AIM      AIM CLI       Running       More..           AF

                "Invalid top line: <top_line>"

         The specified top line for starting the viewport was either out of
         range, or non-numeric.

                "Association between specified window and image already exis

         The specified window is currently associated with <image_name> and t
         AIM prohibits multiple associations between the same window and imag

                "Cannot map another window onto the AIM image"

         The AIM image and window cannot be altered by the user.

ASSOCIATE subtopic? ?

         The ASSOCIATE command maps a specified portion of a window onto an
         image.  Assuming that n = <length> and p = <position>, the ASSOCIATE
         command maps the last n lines of the specified window onto the given
         image starting at the pth line of the image.  For example,

                          ASSOC WIN_1 IM_1 13 12{F4}
_____________________________BOTTOM-OF-SCREEN______________________________
```

Comments: More help

Command: {F4}

```
                                  MORE HELP
 ------------------------------------------------------------------------------
 AIM            AIM       AIM CLI       Running                           AF
    The ASSOCIATE command maps a specified portion of a window onto an
    image. Assuming that n = <length> and p = <position>, the  ASSOCIATE
    command maps the last n lines of the specified window onto the given
    image starting at the pth line of the image.  For example,

                          ASSOC WIN_1 IM_1 13 12

    will map the last 12 lines of WIN_1 onto lines 13..24 of IM_1. This
    forms an abstract association between WIN_1 and IM_1 by creating a
    viewport (lines 13 thru 24) onto which the last 12 lines of WIN_1 are
    mapped. This viewport partitions the image vertically, and its width is
    equal to the width of the image.

    Note: A window may be associated with more than one image at the same
    time; however, a specific window may only be associated with a specific
    image once.

Additional Information Available:

SYNTAX                                    EXAMPLES
COMMAND_PARAMETERS                        ERRORS

ASSOCIATE subtopic? {return}
__________________________BOTTOM-OF-SCREEN_____________________________
```

Comments: The HELP text for ASSOCIATE is redisplayed along with the list of HELP supported subtopics. In this case, only the keyword "Parameters" is supported under the ASSOCIATE command.

Command: {return} - No more information at this level so return to the previous HELP level.

RETURN TO HELP TOP LEVEL

```
AIM                AIM         AIM CLI         Running                           AF
     command maps the last n lines of the specified window onto the given
     image starting at the pth line of the image.  For example,

                             ASSOC WIN_1 IM_1 13 12

     will map the last 12 lines of WIN_1 onto lines 13..24 of IM_1. This
     forms an abstract association between WIN_1 and IM_1 by creating a
     viewport (lines 13 thru 24) onto which the last 12 lines of WIN_1 are
     mapped. This viewport partitions the image vertically, and its width is
     equal to the width of the image.

     Note: A window may be associated with more than one image at the same
     time; however, a specific window may only be associated with a specifi
     image once.

Additional Information Available:

SYNTAX                                  EXAMPLES
COMMAND_PARAMETERS                      ERRORS

ASSOCIATE subtopic?

Topic? CERATE
________________________BOTTOM-OF-SCREEN________________________
```

Comments: The user has returned to the top level of the HELP tree; the "Topic?" prompt reappears.

Command: CERATE - this is an unrecognizable HELP keyword causing a HELP error.

AD-A198151

NAVAL OCEAN SYSTEMS CENTER, SAN DIEGO, CA
APSE INTERACTIVE MONITOR – USER'S MANUAL-ADE™
VERSION BY: TEXAS INSTRUMENTS INC.

4 OF 4
NOSC TD 12
UNCLASSIF
JAN 1988

UNCLASSIFIED

END
DATE
FILMED

HELP ERROR MESSAGE

```
AIM                AIM       AIM CLI        Running       More..            AF
     will map the last 12 lines of WIN_1 onto lines 13..24 of IM_1. This
     forms an abstract association between WIN_1 and IM_1 by creating a
     viewport (lines 13 thru 24) onto which the last 12 lines of WIN_1 are
     mapped. This viewport partitions the image vertically, and its width is
     equal to the width of the image.

     Note: A window may be associated with more than one image at the same
     time; however, a specific window may only be associated with a specific
     image once.

Additional Information Available:

SYNTAX                                    EXAMPLES
COMMAND_PARAMETERS                        ERRORS

ASSOCIATE subtopic? {return}

Topic? CERATE

Sorry, no documentation available on CERATE

Additional Information Available:
{F4}
___________________________BOTTOM-OF-SCREEN_______________________________
```

Comments: More help

Command: {F4}

```
                         MORE HELP
____________________________________________________________________
  AIM             AIM     AIM CLI      Running                 AF

SYNTAX                                  EXAMPLES
COMMAND_PARAMETERS                      ERRORS

ASSOCIATE Subtopic?

Topic? CERATE

Sorry, no documentation available on CERATE

Additional Information Available:

ABORT_AIM                             INFO
ASSOCIATE                             KEYSTROKES
CREATE                                RESET
DEFINE                                RESUME
DELETE                                SCRIPT
DISASSOCIATE                          SET
EXIT                                  SUSPEND
GOTO                                  TERMINATE_EXECUTION
HELP

Topic? CREATE
___________________________BOTTOM-OF-SCREEN_________________________
```

Comments: The "CERATE" keyword was not recognized by HELP. A list of the available HELP topics was displayed, and the "Topic?" prompt reappears.

Command: CREATE - display the HELP information available for the AIM CREATE command.

DISPLAY TOP LEVEL HELP FOR [illegible] COMMAND

```
________________________________________________________________________________
 AIM              AIM    [illegible]                                         AF
Additional Information Available:

ABORT_AIM                                     [illegible]
ASSOCIATE                                     KEYSTROKES
CREATE                                        RESET
DEFINE                                        RESUME
DELETE                                        SCRIPT
DISASSOCIATE                                  SET
EXIT                                          SUSPEND
GOTO                                          TERMINATE_EXECUTION
HELP

Topic? CREATE

CREATE

        The CREATE command allows the user to create a window or an image.

Additional Information Available:

IMAGE                                         WINDOW

CREATE subtopic? *
____________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The HELP text supported for the AIM CREATE command is being displayed.

Command: * - request to display ALL the HELP information in CREATE subtree.

DISPLAY ALL HELP AVAILABLE FOR CREATE COMMAND

```
 AIM               AIM        AIM CLI         Running         More..          AF
ASSOCIATE                                   KEYSTROKES
CREATE                                      RESET
DEFINE                                      RESUME
DELETE                                      SCRIPT
DISASSOCIATE                                SET
EXIT                                        SUSPEND
GOTO                                        TERMINATE_EXECUTION
HELP

Topic? CREATE

CREATE

        The CREATE command allows the user to create a window or an image.

Additional Information Available:

IMAGE                                       WINDOW

CREATE subtopic? *

CREATE
{F4}
____________________________BOTTOM-OF-SCREEN______________________________
```

Comments: More help

Command: {F4}

```
                              MORE HELP
_____________________________________________________________________
AIM            AIM      AIM CLI       Running       More..        AF

Additional Information Available:

IMAGE                                  WINDOW

IMAGE

     The CREATE IMAGE command allows the AIM user to create a new image in
     the AIM environment.  A newly create image will be alphabetically
     entered into the existing image list.

     Note: There is no explicit limit to the number of images that can be
     created during an AIM session.

Additional Information Available:

SYNTAX                         EXAMPLES
COMMAND_PARAMETERS             ERRORS

SYNTAX

              CR*EATE  OBJECT_TYPE => I*MAGE  IMAGE_NAME  =>  <image_name>
{F4}
___________________________BOTTOM-OF-SCREEN__________________________
```

Comments: All the HELP text supported for the AIM CREATE command's subtopics is displayed.

Command: {F4} - display next page of AIM window's output

DISPLAY NEXT PAGE OF AIM WINDOW'S OUTPUT

```
________________________________________________________________________________
 AIM               AIM      AIM CLI       Running       More..          AF

                   OBJECT_TYPE =>  I*MAGE

          Specifies that a new image is to be created in the AIM environment.

                   IMAGE_NAME =>  <image_name>

          Specifies the name of a new image to be created.

   EXAMPLES
          Long  Form:       AIM> CREATE OBJECT_TYPE => IMAGE IMAGE_NAME => IM_1
          Mixed Form:       AIM> CREATE IMAGE IMAGE_NAME => IM_1
          Short Form:       AIM> CR I IM_1

          Each form creates an AIM image named IM_1.

   ERRORS

          The semantic errors associated with this command include:

                  "Image name, <image_name>, already in use"
   {F4}
__________________________BOTTOM-OF-SCREEN_____________________________________
```

Comments: More help

Command: {F4}

```
______________________________________________________________________________
 AIM            AIM      AIM CLI      Running      More..          AF

                "Identifier name is too long"

       The specified image name was longer than 20 characters.

WINDOW

       The CREATE WINDOW command allows the AIM user to create a new window i
       the AIM environment. A newly created image will be alphabetically
       entered into the existing image list.

       Note: There is no explicit limit to the number of windows that can be
       created during an AIM session.

       The default values for the new window's flag settings follow:

               SUSPENDS_OUTPUT_ON_FULL - true,
               INPUT_COMPONENT_ACTIVE - false,
               OUTPUT_COMPONENT_ACTIVE - false,
               OUTPUT_SUSPENDED - false.

Additional Information Available:
{F4}
___________________________BOTTOM-OF-SCREEN______________________________
```

Comments: More help

Command: {F4}

```
  AIM                  AIM        AIM CLI        Running         More..          AF
COMMAND_PARAMETERS                                ERRORS

SYNTAX

                CR*EATE OBJECT_TYPE => W*INDOW WINDOW_NAME =>  <window_name

COMMAND_PARAMETERS

                OBJECT_TYPE =>  W*INDOW

       Specifies that a new window is to be created in the AIM environment.

                WINDOW_NAME =>  <window_name>

       Specifies the name of a new window to be created.

EXAMPLES
       Long Form:        AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WI
       Mixed Form:       AIM> CREATE WINDOW WINDOW_NAME => WIN_1
       Short Form:       AIM> CR W WIN_1

       Each form creates an AIM window named WIN_3.{F4}
___________________________BOTTOM-OF-SCREEN_____________________________


Comments:  More help

Command:  {F4}
```

```
 AIM             AIM        AIM CLI       Running                     AF
       Long Form:        AIM> CREATE OBJECT_TYPE => WINDOW WINDOW_NAME => WIN_
       Mixed Form:       AIM> CREATE WINDOW WINDOW_NAME => WIN_1
       Short Form:       AIM> CR W WIN_1

       Each form creates an AIM window named WIN_3.

ERRORS

       The semantic errors associated with this command include:

               "Window name, <window_name>, already in use"

       The specified window name already exists in the AIM environment.

               "Identifier name is too long"

       The specified window name was longer than 20 characters.

               "Could not create a process for window: <window_name>"

       Access has been denied to disk or disk quota has been exceeded.

CREATE Subtopic? {F10}
___________________________BOTTOM-OF-SCREEN______________________________
```

Comments: All the remaining HELP text supported for the AIM CREATE command's subtopics has been displayed

Command: {F10} - Exit the HELP utility from the present state. This is a keystroke command.

ISSUE AIM EXIT COMMAND

```
 AIM                AIM        AIM CLI        Running                     AF
          Mixed Form:         AIM> CREATE WINDOW WINDOW_NAME => WIN_1
          Short Form:         AIM> CR W WIN_1

          Each form creates an AIM window named WIN_3.

    ERRORS

          The semantic errors associated with this command include:

                  "Window name, <window_name>, already in use"

          The specified window name already exists in the AIM environment.

                  "Identifier name is too long"

          The specified window name was longer than 20 characters.

                  "Could not create a process for window: <window_name>"

          Access has been denied to disk or disk quota has been exceeded.

    CREATE Subtopic? {F10}
    AIM> EXIT
    ___________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The {F10} Switch_to_AIM keystroke command exits HELP and the "AIM>" prompt reappears.

Command: EXIT - Exit the AIM program.

```
                                  EXIT AIM SESSION
_______________________________________________________________________________
       Mixed Form:        AIM> CREATE WINDOW WINDOW_NAME => WIN_1
       Short Form:        AIM> CR W WIN_1

       Each form creates an AIM window named WIN_3.

ERRORS

       The semantic errors associated with this command include:

               "Window name, <window_name>, already in use"

       The specified window name already exists in the AIM environment.

               "Identifier name is too long"

       The specified window name was longer than 20 characters.

               "Could not create a process for window: <window_name>"

       Access has been denied to disk or disk quota has been exceeded.

CREATE Subtopic? {F10}
AIM> EXIT
-)
__________________________BOTTOM-OF-SCREEN_____________________________
```

Comments: The AIM has terminated its execution and the prompt from the underlying APSE reappears.

Command: None.

A.7 SESSION 7 - AIM INFO UTILITY

The AIM INFO Utility session explains the use of the INFO command. The INFO command gives the user information on the current status of associations, keystrokes, and mode settings. The utility has successive levels of information and is based on the same format as the HELP utility. The following text files are used in this session:

1. demo1.aim

```
CREATE  IMAGE   IMAGE_1
CREATE  IMAGE   IMAGE_2
CREATE  WINDOW  WINDOW_A
CREATE  WINDOW  WINDOW_B
CREATE  WINDOW  WINDOW_C
CREATE  WINDOW  WINDOW_D
ASSOC  WINDOW_A  IMAGE_1  1   8
ASSOC  WINDOW_B  IMAGE_1  9   8
ASSOC  WINDOW_C  IMAGE_1  17  8
ASSOC  WINDOW_C  IMAGE_2  1   12
ASSOC  WINDOW_D  IMAGE_2  13  12
```

INVOKE AIM FROM APSE

```
-) AIM
















______________________________BOTTOM-OF-SCREEN______________________________
```

Comments:

Command: AIM - invoke the AIM from the APSE.

SWITCH TO AIM IMAGE

```
 MAIN               MAIN           FRCH:016         Running                          AF

AOS/VS CLI REV 05.01.00.00         9-APR-85      13:26:43 {F10}













_____________________________________BOTTOM-OF-SCREEN______________________________________
```

Comments: The AIM initializes the current image to be MAIN. To enter AIM textual command the user must communicate through the AIM image to the AIM command interpreter.

Command: {F10} - Set the current image to be the AIM image.

INVOKE DEMO1 SCRIPT

```
 AIM            AIM       AIM CLI       Running                          AF
AIM> SCRIPT "demol.aim"









___________________________BOTTOM-OF-SCREEN________________________________
```

Comments: The AIM image is initially blank.

Command: SCRIPT "demol.aim" invoke the execution of a command script. The script resides in the file DEMO1.

ECHO SCRIPT LINES

```
 AIM              AIM       AIM CLI        Running                        AF
AIM> SCRIPT "demol.aim"
AIM> CREATE  IMAGE  IMAGE_1
AIM> CREATE  IMAGE  IMAGE_2
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> CREATE  WINDOW  WINDOW_D
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> ASSOC  WINDOW_C  IMAGE_2  1   12
AIM> ASSOC  WINDOW_D  IMAGE_2  13  12
AIM> INFO







_____________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The AIM command interpreter has echoed each command line of the script as it was being executed.

Command: INFO - invoke AIM INFO utility.

THE AIM INFO UTILITY

```
 AIM               AIM       AIM CLI        Running                      AF
AIM> SCRIPT "demol.aim"
AIM> CREATE  IMAGE   IMAGE_1
AIM> CREATE  IMAGE   IMAGE_2
AIM> CREATE  WINDOW  WINDOW_A
AIM> CREATE  WINDOW  WINDOW_B
AIM> CREATE  WINDOW  WINDOW_C
AIM> CREATE  WINDOW  WINDOW_D
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> ASSOC  WINDOW_C  IMAGE_2  1   12
AIM> ASSOC  WINDOW_D  IMAGE_2  13  12
AIM> INFO

Info is available for the following AIM objects:

IMAGES
KEYSTROKES
TERMINAL
WINDOWS

INFO Object?  IM
___________________________BOTTOM-OF-SCREEN___________________________
```

Comments: The INFO utility lists ALL the AIM objects for which information is available. When INFO prompts for an object name, key in one of the AIM keywords from the list.

Command: IM - The "IM" is an valid abbreviation for the INFO keyword "IMAGES". The command requests an alphabetical listing of ALL the current images in the AIM environment.

```
                         ALPHABETICAL IMAGE NAME LIST
___________________________________________________________________________
 AIM              AIM      AIM CLI         Running      More..          AF
AIM> CREATE   IMAGE   IMAGE_1
AIM> CREATE   IMAGE   IMAGE_2
AIM> CREATE   WINDOW  WINDOW_A
AIM> CREATE   WINDOW  WINDOW_B
AIM> CREATE   WINDOW  WINDOW_C
AIM> CREATE   WINDOW  WINDOW_D
AIM> ASSOC  WINDOW_A  IMAGE_1  1   8
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> ASSOC  WINDOW_C  IMAGE_2  1   12
AIM> ASSOC  WINDOW_D  IMAGE_2  13  12
AIM> INFO

Info is available for the following AIM objects:

IMAGES
KEYSTROKES
TERMINAL
WINDOWS

INFO Object? IM

The current AIM images include: {F4}
___________________________BOTTOM-OF-SCREEN______________________________
```

Comments: More Info

Command: {F4}

MORE INFO

```
 AIM            AIM      AIM CLI      Running                    AF
AIM> ASSOC  WINDOW_B  IMAGE_1  9   8
AIM> ASSOC  WINDOW_C  IMAGE_1  17  8
AIM> ASSOC  WINDOW_C  IMAGE_2  1   12
AIM> ASSOC  WINDOW_D  IMAGE_2  13  12
AIM> INFO

Info is available for the following AIM objects:

IMAGES
KEYSTROKES
TERMINAL
WINDOWS

INFO Object? IM

The current AIM images include:

     AIM
     IMAGE_1
     IMAGE_2
     MAIN

INFO Image Name?  MAIN
___________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The INFO utility lists ALL active images and then prompts the user to see if more information is requested about a specific image.

Command: MAIN - request for the available information on the image MAIN.

```
                                INFO ON MAIN
 ____________________________________________________________________________
 AIM           AIM       AIM CLI      Running                          AF
IMAGES
KEYSTROKES
TERMINAL
WINDOWS

INFO Object? IM

The current AIM images include:

    AIM
    IMAGE_1
    IMAGE_2
    MAIN

INFO Image Name? MAIN

Image Name:  MAIN

Window Associations:
Window Name          Top Line          Viewport Length
MAIN                    1                    24

INFO Image Name? AIM
___________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The INFO utility now displays ALL the information available for MAIN. INFO prompts the user for another image name.

Command: AIM - Request info on AIM.

```
                               INFO ON AIM
___________________________________________________________________________
 AIM             AIM       AIM CLI        Running                     AF

     AIM
     IMAGE_1
     IMAGE_2
     MAIN

INFO Image Name? MAIN

Image Name:  MAIN

Window Associations:
Window Name          Top Line          Viewport Length
MAIN                    1                     24

INFO Image Name? AIM

Image Name:  AIM

Window Associations:
Window Name          Top Line          Viewport Length
AIM                     1                     24

INFO Image Name? IMAGE_1
___________________________BOTTOM-OF-SCREEN_____________________________
```

Comments: Now displayed is all the information available for AIM. Again INFO prompts the user for another image name.

Command: IMAGE_1

```
                              INFO ON IMAGE_1
________________________________________________________________________
 AIM           AIM      AIM CLI       Running      More..          AF
     AIM
     IMAGE_1
     IMAGE_2
     MAIN

INFO Image Name? MAIN

Image Name:  MAIN

Window Associations:
Window Name         Top Line         Viewport Length
MAIN                    1                  24

INFO Image Name? AIM

Image Name:  AIM

Window Associations:
Window Name         Top Line         Viewport Length
AIM                     1                  24

INFO Image Name? IMAGE_1
{F4}
__________________________BOTTOM-OF-SCREEN______________________________
```

Comments: More info

Command: {F4}

MORE INFO

```
________________________________________________________________________________
 AIM               AIM       AIM CLI        Running                          AF
Window Associations:
Window Name          Top Line           Viewport Length
MAIN                    1                      24

INFO Image Name? AIM

Image Name:  AIM

Window Associations:
Window Name          Top Line           Viewport Length
AIM                     1                      24

INFO Image Name? IMAGE_1

Image Name:  IMAGE_1

Window Associations:
Window Name          Top Line           Viewport Length
WINDOW_A                1                      8
WINDOW_B                9                      8
WINDOW_C                17                     8

INFO Image Name?  {return}
___________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The INFO utility displays ALL the information available for IMAGE_1. Note that the prompt is for another image name as there are NO further sublevels of information available for images. In this case, ALL the available information for IMAGE_1 has been displayed; INFO prompts the user for another image name.

Command: {return} - return to the previous INFO level.

RETURN TO TOP LEVEL OF INFO TREE

```
 AIM               AIM      AIM CLI       Running                        AF
Window Name          Top Line          Viewport Length
MAIN                    1                       24

INFO Image Name? AIM

Image Name:  AIM

Window Associations:
Window Name          Top Line          Viewport Length
AIM                     1                       24

INFO Image Name? IMAGE_1

Image Name:  IMAGE_1

Window Associations:
Window Name          Top Line          Viewport Length
WINDOW_A                1                      8
WINDOW_B                9                      8
WINDOW_C               17                      8

INFO Image Name?
INFO Object? KEYS
______________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The carriage return in response to the "INFO Image Name?" prompt caused the "INFO Object?" to reappear. The current INFO level is now the top level.

Command: KEYS - list ALL keystroke command names alphabetically.

LIST KEYSTROKE COMMAND NAMES ALPHABETICALLY

```
______________________________________________________________________________
 AIM               AIM       AIM CLI       Running       More..           AF

Window Associations:
Window Name           Top Line           Viewport Length
WINDOW_A               1                      8
WINDOW_B               9                      8
WINDOW_C               17                     8

INFO Image Name?
INFO Object? KEYS

AIM keystroke names include:

ABORT_SCRIPT
CLEAR_WINDOW
NEXT_IMAGE
NEXT_PAGE
NEXT_VIEWPORT
PREVIOUS_IMAGE
PREVIOUS_VIEWPORT
REDISPLAY_SCREEN
RETURN_TO_PREV_IMAGE
SWITCH_TO_AIM
{F4}
______________________________BOTTOM-OF-SCREEN________________________________
```

Comments: More info

Command: {F4}

```
                                MORE INFO
 ____________________________________________________________________________
  AIM            AIM       AIM CLI       Running                          AF
Window Associations:
Window Name          Top Line           Viewport Length
WINDOW_A                1                    8
WINDOW_B                9                    8
WINDOW_C               17                    8

INFO Image Name?
INFO Object? KEYS

AIM keystroke names include:

ABORT_SCRIPT
CLEAR_WINDOW
NEXT_IMAGE
NEXT_PAGE
NEXT_VIEWPORT
PREVIOUS_IMAGE
PREVIOUS_VIEWPORT
REDISPLAY_SCREEN
RETURN_TO_PREV_IMAGE
SWITCH_TO_AIM

INFO Keystroke Name?  *
__________________________BOTTOM-OF-SCREEN__________________________
```

Comments: INFO displays an alphabetical listing of ALL the AIM keystroke command names and asks the user if more info is desired about a specific keystroke.

Command: * - list ALL keystroke command to key sequence associations.

LIST ALL KEYSTROKE TO KEY SEQUENCE BINDINGS

```
_____________________________________________________________________________
 AIM            AIM       AIM CLI       Running                         AF
NEXT_VIEWPORT
PREVIOUS_IMAGE
PREVIOUS_VIEWPORT
REDISPLAY_SCREEN
RETURN_TO_PREV_IMAGE
SWITCH_TO_AIM

INFO Keystroke Name? *

Keystroke Command Name       Key Sequence

ABORT_SCRIPT                    F1
CLEAR_WINDOW                    F2
NEXT_IMAGE                      F3
NEXT_PAGE                       F4
NEXT_VIEWPORT                   F5
PREVIOUS_IMAGE                  F6
PREVIOUS_VIEWPORT               F7
REDISPLAY_SCREEN                F8
RETURN_TO_PREVIOUS_IMAGE        F9
SWITCH_TO_AIM                   F10

INFO Keystroke Name?  {return}
___________________________BOTTOM-OF-SCREEN______________________________
```

Comments: INFO displays an alphabetical listing of ALL the AIM's keystroke command names with their associated key sequence.

Command: {return} - return to previous INFO level.

RETURN TO PREVIOUS INFO LEVEL

```
 AIM              AIM        AIM CLI       Running                        AF
PREVIOUS_IMAGE
PREVIOUS_VIEWPORT
REDISPLAY_SCREEN
RETURN_TO_PREV_IMAGE
SWITCH_TO_AIM

INFO Keystroke Name? *

Keystroke Command Name       Key Sequence

ABORT_SCRIPT                   F1
CLEAR_WINDOW                   F2
NEXT_IMAGE                     F3
NEXT_PAGE                      F4
NEXT_VIEWPORT                  F5
PREVIOUS_IMAGE                 F6
PREVIOUS_VIEWPORT              F7
REDISPLAY_SCREEN               F8
RETURN_TO_PREVIOUS_IMAGE       F9
SWITCH_TO_AIM                  F10

INFO Keystroke Name?
INFO Object?  TERMINAL
___________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The "INFO Object?" prompt reappears.

Command: TERMINAL - request for the information available on the terminal.

RETURN TO PREVIOUS INFO LEVEL

```
 AIM                AIM       AIM CLI       Running                         AF
SWITCH_TO_AIM

INFO Keystroke Name? *

Keystroke Command Name        Key Sequence

ABORT_SCRIPT                    F1
CLEAR_WINDOW                    F2
NEXT_IMAGE                      F3
NEXT_PAGE                       F4
NEXT_VIEWPORT                   F5
PREVIOUS_IMAGE                  F6
PREVIOUS_VIEWPORT               F7
REDISPLAY_SCREEN                F8
RETURN_TO_PREVIOUS_IMAGE        F9
SWITCH_TO_AIM                   F10

INFO Keystroke Name?
INFO Object? TERMINAL

Terminal Type: UNDEFINED

INFO Object?  WINDOW MAIN
___________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The "INFO Object?" prompt reappears. Note the terminal type will be the name of the terminal being used.

Command: WINDOW MAIN - request for ALL the information available on the window MAIN. Note that response can be combined.

```
                              WINDOW MAIN INFO
____________________________________________________________________________
 AIM            AIM       AIM CLI       Running       More..            AF

Keystroke Command Name       Key Sequence

ABORT_SCRIPT                   F1
CLEAR_WINDOW                   F2
NEXT_IMAGE                     F3
NEXT_PAGE                      F4
NEXT_VIEWPORT                  F5
PREVIOUS_IMAGE                 F6
PREVIOUS_VIEWPORT              F7
REDISPLAY_SCREEN               F8
RETURN_TO_PREVIOUS_IMAGE       F9
SWITCH_TO_AIM                  F10

INFO Keystroke Name?
INFO Object? TERMINAL

Terminal Type: UNDEFINED

INFO Object? WINDOW MAIN

Window name: MAIN
{F4}
__________________________BOTTOM-OF-SCREEN__________________________________
```

Comments: More info

Command: {F4}

```
                                   MORE INFO
_________________________________________________________________________
 AIM            AIM       AIM CLI       Running                     AF
Input Pad Name: NONE
Output Pad Name: NONE

Pad Mode:  NONE
Suspend Output on Full Window:  ON
Suspend Output on Switch-to-AIM:  ON
Window Output Status:  ACTIVE

Process: FRCH:016

Image Associations:
Image  Name          Top Line          Viewport Length
MAIN                    1                     24


Information is available for:

ASSOCIATIONS
MODES
PADS
PROCESS_STATUS

INFO Window MAIN Sub-block? {return}
___________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The information on the window MAIN is listed. The user is given the option for more info at lower levels (though not entered here).

Command: {return} - return to the previous level.

RETURN TO PREVIOUS INFO LEVEL

```
_________________________________________________________________________
 AIM            AIM      AIM CLI       Running                       AF
Output Pad Name: NONE

Pad Mode:  NONE
Suspend Output on Full Window:  ON
Suspend Output on Switch-to-AIM:  ON
Window Output Status:  ACTIVE

Process: FRCH:016

Image Associations:
Image  Name          Top Line          Viewport Length
MAIN                    1                     24


Information is available for:

ASSOCIATIONS
MODES
PADS
PROCESS_STATUS

INFO Window MAIN Sub-block?
INFO Window name?  WINDOW_A
__________________________BOTTOM-OF-SCREEN_____________________________
```

Comments: "The INFO Window Name?" now appears. Now request information on another window.

Command: WINDOW_A - Request ALL information on WINDOW_A.

DISPLAY ALL INFORMATION ON WINDOW_A

```
________________________________________________________________________
 AIM            AIM      AIM CLI       Running      More..          AF
Output Pad Name: NONE

Pad Mode:  NONE
Suspend Output on Full Window:  ON
Suspend Output on Switch-to-AIM:  ON
Window Output Status:  ACTIVE

Process: FRCH:016

Image Associations:
Image  Name          Top Line         Viewport Length
MAIN                   1                    24


Information is available for:

ASSOCIATIONS
MODES
PADS
PROCESS_STATUS

INFO Window MAIN Sub-block?
INFO Window name?  WINDOW_A {F4}
__________________________BOTTOM-OF-SCREEN_____________________________
```

Comments: More info

Command: {F4}

```
                                MORE INFO
 _____________________________________________________________________________
  AIM            AIM       AIM CLI        Running       More..            AF
 Window Name: WINDOW_A

 Input Pad Name: NONE
 Output Pad Name: NONE

 Pad Mode:  NONE
 Suspend Output on Full Window:  ON
 Suspend Output on Switch-to-AIM:  ON
 Window Output Status:  ACTIVE

 Process: FRCH:017

 Image Associations:
 Image  Name          Top Line          Viewport Length
 IMAGE_1                1                      8


 Information is available for:

 ASSOCIATIONS
 MODES
 PADS
 PROCESS_STATUS {F4}
 ___________________________BOTTOM-OF-SCREEN_____________________________
```

Comments: More info

Commands: {F4}

```
                                MORE INFO
___________________________________________________________________________
 AIM            AIM       AIM CLI       Running                       AF
Input Pad Name: NONE
Output Pad Name: NONE

Pad Mode:  NONE
Suspend Output on Full Window:  ON
Suspend Output on Switch-to-AIM:  ON
Window Output Status:  ACTIVE

Process: FRCH:017

Image Associations:
Image  Name          Top Line          Viewport Length
IMAGE_1                 1                     8


Information is available for:

ASSOCIATIONS
MODES
PADS
PROCESS_STATUS

INFO Window WINDOW_A Sub-block?  {return}
___________________________BOTTOM-OF-SCREEN______________________________
```

Comments: ALL the information for the window WINDOW_A is displayed. Notice that the text is naturally broken into 4 sub-blocks of information:

1. ASSOCIATIONS,

2. MODES,

3. PADS,

4. PROCESS_STATUS,

The user of this utility can display any one, or all of these sub-blocks

Command: {return} - return to previous INFO level.

RETURN TO PREVIOUS INFO LEVEL

```
______________________________________________________________________________
 AIM              AIM        AIM CLI         Running                        AF
Output Pad Name: NONE

Pad Mode:  NONE
Suspend Output on Full Window:  ON
Suspend Output on Switch-to-AIM:  ON
Window Output Status:  ACTIVE

Process:  FRCH:017

Image Associations:
Image  Name          Top Line          Viewport Length
IMAGE_1                 1                         8


Information is available for:

ASSOCIATIONS
MODES
PADS
PROCESS_STATUS

INFO Window WINDOW_A Sub-block?
INFO Window Name? WINDOW_C ASSOC
___________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The "INFO Window Name?" reappears allowing the user to acquire information for another window.

Command: WINDOW_C ASSOC - "ASSOC" uniquely identifies "ASSOCIATIONS". This command has two INFO keywords, "WINDOW_C" and "ASSOC". It will display ONLY the "ASSOCIATIONS" for the window WINDOW_C.

DISPLAY ONLY WINDOW_C'S ASSOCIATION INFORMATION

```
 AIM            AIM            Running                                   AF
Image Associations:
Image  Name          Top Line         Viewport Length
IMAGE_1                1                     8


Information is available for:

ASSOCIATIONS
MODES
PADS
PROCESS_STATUS

INFO Window WINDOW_A Sub-block?
INFO Window Name? WINDOW_C ASSOC

Window Name:  WINDOW_C

Image Associations:
Image  Name          Top Line         Viewport Length
IMAGE_1               17                     8
IMAGE_2                1                    12

INFO Window WINDOW_C Sub-block? MODSE
___________________________BOTTOM-OF-SCREEN______________________________
```

Comments: ONLY the Image associations information was displayed for WINDOW_C. Since there are other sub-blocks of information available for the window, INFO prompts the user for another sub-block name.

Command: MODSE - Invalid INFO keyword. Should have been "MODES".

EXIT INFO UTILITY

```
_____________________________________________________________________________
 AIM              AIM       AIM CLI       Running                         AF

INFO Window WINDOW_A Sub-block?
INFO Window Name? WINDOW_C ASSOC

Window Name:  WINDOW_C

Image Associations:
Image  Name         Top Line          Viewport Length
IMAGE_1               17                     8
IMAGE_2               1                      12

INFO Window WINDOW_C Sub-block? MODSE

Unrecognized INFO keyword, MODSE.

Information is available for:

ASSOCIATIONS
MODES
PADS
PROCESS_STATUS

INFO Window WINDOW_C Sub-block?  {F10}
___________________________BOTTOM-OF-SCREEN______________________________
```

Comments: The {F10} keystroke command exits INFO and the "AIM>" prompt reappears.

Command: {F10}

ENTER EXIT COMMAND

```
 AIM            AIM       AIM CLI       Running                            AF
INFO Window WINDOW_A Sub-block?
INFO Window Name? WINDOW_C ASSOC

Window Name:  WINDOW_C

Image Associations:
Image  Name         Top Line         Viewport Length
IMAGE_1                17                   8
IMAGE_2                1                    12

INFO Window WINDOW_C Sub-block? MODSE

Unrecognized INFO keyword, MODSE.

Information is available for:

ASSOCIATIONS
MODES
PADS
PROCESS_STATUS

INFO Window WINDOW_C Sub-block?
AIM> EXIT
___________________________BOTTOM-OF-SCREEN_____________________________
```

Comments: Enter the EXIT command

Command: EXIT - Exit the AIM program.

EXIT THE AIM

```
INFO Window WINDOW_A Sub-block?
INFO Window Name? WINDOW_C ASSOC

Window Name:  WINDOW_C

Image Associations:
Image  Name          Top Line          Viewport Length
IMAGE_1                17                   8
IMAGE_2                1                    12

INFO Window WINDOW_C Sub-block? MODSE

Unrecognized INFO keyword, MODSE.

Information is available for:

ASSOCIATIONS
MODES
PADS
PROCESS_STATUS

INFO Window WINDOW_C Sub-block?
AIM> EXIT
-)
__________________________BOTTOM-OF-SCREEN____________________________
```

Comments: The AIM has terminated its execution and the prompt from the underlying APSE reappears.

Command: None.

APPENDIX B

REFERENCES

[DOD83] United States Department of Defense, "Reference Manual for the Ada Programming Language Draft, Revised MIL-STD-1815A," February 1983.

[ANSI77] American National Standards Institute, "American National Standard Code for Information Interchange (ANSI Standard X3.4-1977)," June 1977.

[ANSI79] American National Standards Institute, "American National Standard Additional Controls for Use with American National Standard Code for Information Interchange (ANSI Standard X3.64-1979)," July 1979.

[DEC82] Digital Equipment Corporation, "VAX-11 Utilities Reference Manual - Help Libraries", Maynard, MA, May 1982.

[NYU81] Charles, Philippe and Fisher, Gerald, "Using the NYU Parser Generator," Courant Institute, New York University, New York, 1981.

INDEX